

Honeywell Bull

TIME-SHARING
APPLICATIONS LIBRARY GUIDE
VOLUME III - INDUSTRY
ADDENDUM A

SERIES 600/6000

APPLICATIONS

SUBJECT:

Additional and Revised Time-Sharing Programs.

SPECIAL INSTRUCTIONS:

This update is the first addendum to DA45, Revision 2, dated December 1972. Sixteen new programs and nine revised programs are being added with this addendum; these programs are listed in the revised Preface.

Insert the attached pages into the manual as indicated in the collating instructions on the back of this cover. Change bars in the margins indicate new and changed information; asterisks denote deletions.

NOTE: This cover should be inserted following the manual cover to indicate the updating of the document with Addendum A.

DATE:

May 1973

ORDER NUMBER:

DA45A, Rev. 2

Printed in France

Ref : 19.53.108 A1

COLLATING INSTRUCTIONS

To update the manual, remove old pages and insert new pages as follows:

<u>Remove</u>	<u>Insert</u>
Title/Preface	Title/Preface
iii/iv	iii/iv
vii through xii	vii through xiii, blank
MS-11 through MS-16	MS-11 through MS-16
	MS-16.1 through MS-16.3, blank
MS-23 through MS-26	MS-23 through MS-26
MS-39 through MS-42	MS-39 through MS-42
MS-43 through MS-48	MS-43 through MS-48
	MS-54.1/MS-54.2
MS-61 through MS-63	MS-61 through MS-63, blank
	MS-86.1 through MS-86.5, blank
MS-91 through MS-99	MS-91 through MS-100
	GP-6.1 through GP-6.3, blank
ED-1/ED-2	ED-1 through ED-2.1, blank
ED-3/ED-4	ED-3/ED-4
ED-5/ED-6	ED-5/ED-6
DE-3/DE-4	DE-3/DE-4
	DE-4.1 through DE-4.3, blank
	UM-0.1 through UM-02
	UM-2.1, blank
	UM-2.3 through UM-2.4
	UM-2.5 through UM-2.7, blank
	UM-2.9/UM-2.10
	UM-2.11/UM-2.12
	UM-10.1/UM-10.2
	UM-10.3/blank
	UM-12.1 through UM-12.3, blank
	UM-12.5, blank
	UM-23, blank

In addition, several existing programs are revised to run on Series 6000 FORTRAN. These programs include:

CPM	TCAST
GASPIIA	DRIVES
INT01	EXPERn
INTLP	PREPRS
LNPROG	

A complete printing of the programs in the library is available by listing the library program, CATALOG. A copy of this listing follows the Contents.

Other Series 600/6000 Time-Sharing Library programs are described in the following documents:

Series 600/6000 Time-Sharing Applications Library Guide, Volume I - Mathematics, Order Number DA43

Series 600/6000 Time-Sharing Applications Library Guide, Volume II - Statistics, Order Number DA44

Series 600/6000 Time-Sharing Applications Library Guide, Volume IV - Business and Finance, Order Number DA46

Series 600/6000 Time-Sharing Applications Library programs are also available to users of the DATANETWORK service. Please contact your local Honeywell representative for further details.

This document describes programs that originated from a variety of sources, such as users and the Honeywell field organization. The programs and documentation are made available in the general form and degree of completeness in which they were received. Honeywell Information Systems Inc., therefore, neither guarantees the accuracy of the programs nor assumes support responsibility.

CONTENTS

	Page
 MANAGEMENT SCIENCE AND OPTIMIZATION (MS)	
ASIGNIT The Assignment Problem	MS-1
COEFS Determines Seasonal Coefficients of an Observation Series of Two Cycles	MS-3
COMBI Determines Economic Order Quantity For A Group of Items — Different Discounts	MS-5
CPM Solves Critical Path Method Problems	MS-11
CSM Non-negative Vector X to Maximize Convex Function F (X)	MS-17
DAVIDON Davidon's Unconstrained Optimization	MS-21
GASPIIA A FORTRAN-Based Simulation Language	MS-23
GEOSIM Schedules Machine Shop Jobs Using Heuristic Geometric Approach	MS-27
GPROG Geometric Programming	MS-31
INTOI Zionts' Modification of Balas' Routine for 0-1 Integer Programming	MS-39
INTLP Gomory's Pure and Mixed Integer Programming	MS-43
JSSIM Job Shop Scheduling	MS-49
KILTER "Out of Kilter" Algorithm for Minimum Cost Circulation Network Problem	MS-53
LINPRO Linear Programming (18 x 30 Maximum Size)	MS-55
LNPROG Linear Programming (30 x 50 Maximum Size)	MS-61
LOGIC3 Unconstrained Non-Linear Optimization	MS-65
MAXFLOW Finds the Maximum Flow Through a Network	MS-69
MAXOPT Unconstrained Non-Linear Optimization	MS-71
OPTIM Optimum Service Level For One Inventory Item	MS-75
PERT Performs a Simple PERT Network Analysis	MS-81
SHORTEST Calculates Shortest Path — Minimum Spanning Tree	MS-85
SMOOTH Calculates a Smoothed Series	MS-87
TCAST Performs Time Series Forecasting	MS-91
TRANSP0 An Algorithm to Solve Transportation Problems	MS-101
UNDEQ Solutions for a System of Equations	MS-107
 ENGINEERING (EN)	
ACNET Calculates Gain and Phase of Linear Circuit	EN-1
BEMDES Selects Steel Beams for Various Loads and Supports	EN-11

CATALOG OF SERIES 6000/600 T-S LIBRARY PROGRAMS

FILE TYPE INDICATOR:

LANGUAGE (FIRST LETTER)	MODE (FOLLOWING LETTERS)
A ALGOL	P (OR BLANK) PROGRAM
B BASIC	S SUBROUTINE(S)
C CARDIN	F FUNCTION(S)
D DATABASIC	P-S PROGRAM WITH EXTRACTABLE SUBROUTINE(S)
E TEXT EDITOR	R RELOCATABLE OBJECT (C*)
F FORTRAN	H SYSTEM LOADABLE OBJECT (H*)
G GMAP	L USER'S RANDOM LIBRARY

ALL FILES ARE SOURCE MODE UNLESS OTHERWISE INDICATED.

SUBJECTS

DOCUMENTATION MANUAL

MATHEMATICS (MA)ORDER # DA43
 INTEGRATION
 DIFFERENTIATION, DIFFERENTIAL EQ.
 INTERPOLATION
 POLYNOMIALS
 LINEAR EQUATIONS
 MATRICES
 NON-LINEAR EQUATIONS
 SPECIAL FUNCTION EVALUATION
 LOGIC AND NUMBER THEORY
 STATISTICS (ST)ORDER # DA44
 CURVE FITTING AND REGRESSION
 ANALYSIS OF VARIANCE
 PROBABILITY DISTRIBUTIONS
 CONFIDENCE LIMITS
 HYPOTHESIS TESTING
 DESCRIPTIVE STATISTICS
 RANDOM NUMBER GENERATION
 MISCELLANEOUS STATISTICS
 MANAGEMENT SCIENCE AND OPTIMIZATION (MS)ORDER # DA45
 LINEAR PROGRAMMING
 INTEGER PROGRAMING
 NON-LINEAR OPTIMIZATION
 NETWORK ANALYSIS
 FORECASTING
 SIMULATION
 ENGINEERING (EN)
 GEOMETRIC AND PLOTTING (GP)
 EDUCATION AND TUTORIAL (ED)
 DEMONSTRATION (DE)
 UTILITY AND MISCELLANEOUS (UM)
 BUSINESS AND FINANCE (BF)ORDER # DA46

THE DOCUMENTATION FOR THESE PROGRAMS IS AVAILABLE IN FOUR MANUALS:
 SEE ORDER # DA43 FOR PROGRAMS IN MATHEMATICS
 ORDER # DA44 FOR PROGRAMS IN STATISTICS
 ORDER # DA46 FOR PROGRAMS IN BUSINESS AND FINANCE
 ORDER # DA45 FOR PROGRAMS IN ALL OTHER CATEGORIES.

SUBROUTINES THAT ARE CALLED BY A PROGRAM AND MUST BE EXECUTED WITH IT
 ARE LISTED IN BRACKETS AT THE END OF THE DESCRIPTION.

THESE PROGRAMS HAVE ALL BEEN REVIEWED AND TESTED BUT NO RESPONSIBILITY
 CAN BE ASSUMED.

*****MA--MATHEMATICS*****

INTEGRATION

CLCINT	FF	INTEGRATION BY SIMPSON'S RULE
FINT	FF	EVALUATE FOURIER INTEGRALS BY FILON'S FORMULA
GAHER	FF	GAUSS-HERMITE QUADRATURE
GALA	FF	GAUSS-LAGUERRE QUADRATURE
GAUSSN	FF	EVALUATE DEFINITE DOUBLE OR TRIPLE INTEGRALS
GAUSSQ	FF	GAUSSIAN QUADRATURE
NC0ATES	FP-S	NEWTON-C0ATES QUADRATURE
NUMINT	B	GAUSSIAN QUADRATURE
R0MBINT	FP-S	R0MBERG INTEGRATION
SPLINE	B	INTEGRATE TABULATED FUNCTION BY SPLINE FITS

DIFFERENTIATION, DIFFERENTIAL EQ.

AMPBX	FS	ADAMS-M0ULTON FOR 1ST-ORDER DIFF. EQNS (RKPBX)
FDRVUL	FF	DIFFERENTIATE TABULATED FUNCTION, UNEQUAL SPACING
HDRVEB	FF	DIFFERENTIATE TABULATED FUNCTION, EQUAL SPACING
RKPBX	FS	RUNGE-KUTTA FOR 1ST-ORDER DIFF. EQNS

***INTERPOLATION*

	**	
SPLINT	B	SPLINE INTERPOLATION
TNT1	FF	SINGLE LAGRANGIAN INTERPOLATION (TLU1)
TNT2	FF	DOUBLE LAGRANGIAN INTERPOLATION (TLU1)
TNT2A	FF	VARIABLE DOUBLE LINEAR INTERPOLATION (TLU1)

POLYNOMIALS

BIC0F	FS	CALCULATE BINOMIAL COEFFICIENTS
CLPLY	FF	EVALUATE REAL POLY AT REAL ARGUMENT
CP0LY	FS	FINDS ZEROES OF A COMPLEX POLYNOMIAL
CP0LY-DR	FP	FINDS ZEROES OF A COMPLEX POLYNOMIAL (CP0LY)
DVALG	FS	POLYNOMIAL DIVISION
EUALG	FS	G.C.D. OF TWO POLYNOMIALS (DVALG)
MTALG	FS	MULTIPLY POLYNOMIALS
PLMLT	FS	REAL POLY COEFFICIENTS RECONSTRUCTED FROM REAL ROOTS
P0LRTS	FP	SOLUTION OF POLY BY BAIRSTOWS METHOD
P0LYC	FS	REAL POLY COEFFICIENTS RECONSTRUCTED FROM COMPLEX ROOTS
P0LYV	FS	EVALUATE REAL POLY AT COMPLEX ARGUMENT
QUADEQ	B	SOLUTION TO QUADRATIC EQUATIONS
R00TER	B	SOLUTION OF POLY BY BAIRSTOWS METHOD
ZC0P	FP	ROOTS OF POLYNOMIAL WITH COMPLEX COEFF.
ZC0P2	FS	ROOTS OF POLYNOMIAL WITH COMPLEX COEF. (ZC0P2)
Z0RP	FP	ROOTS OF REAL POLY
Z0RP2	FS	ROOTS OF REAL POLY

LINEAR EQUATIONS

GJSIMEQ	FS	SOLVE LINEAR SYSTEMS BY GAUSS-JORDAN
GSEIDEL	FP-S	SOLVE LINEAR SYSTEMS BY GAUSS-SEIDEL
LINEQ	FS	SOLVE LINEAR SYSTEMS BY GAUSSIAN ELIMINATION
LINSR	FP	SOLVE LINEAR SYSTEMS BY GAUSSIAN ELIMINATION (LINEQ)
SIMEQN	B	SOLVE LINEAR SYSTEMS BY MATRIX INVERSION

MATRICES

DETE	FF	EVALUATE DETERMINANT OF REAL MATRIX
D0MEIG	FP-S	D0MINANT EIGENVALUES OF REAL MATRIX
EIG1	FS	EIGENVALUES OF SYM MATRIX BY JAC0BI METHOD
EIGNHC	FS	EIGENVALUES & VECTORS OF COMPLEX NON-HERMITIAN MATRICES
EIGNSR	FS	EIGENVALUES & VECTORS OF REAL NON-SYMMETRIC MATRICES
EIGSR	FP	EIGENVALUES AND VECTORS OF REAL SYM. MATRIX (EIG1)
LINSO	FS	SOLVE LIN. SYS. W/ SYMMETRIC DOUBLE PREC. COEF. MATRIX
LINSS	FS	SOLVE LIN. SYS. W/ SYMMETRIC SINGLE PREC. COEF. MATRIX
MTINV	FS	MATRIX INVERSION BY PIVOTS
MTMPY	FS	MATRIX MULTIPLICATION
MTRAN	FS	TRANSP0SE A MATRIX
SPEIG1	FS	SPECIAL EIGEN PROBLEMS (EIG1)
SYMEIG	FP	EIGENVALUES OF SYM MATRIX BY JAC0BI METHOD

NON-LINEAR EQUATIONS

BROWN	FS	SOLN OF SIMULTANEOUS SYSTEMS BY BROWN METHOD
SECANT	FS	SOLN OF SIMULTANEOUS SYSTEMS BY SECANT METHOD (MTINV)
SOLN	FF	ZERO OF AN ARBITRARY FUNCTION
ZEROES	B	ZERO, MAX, MIN OF FUNCTION

***SPECIAL FUNCTION EVALUATION*

**

ARCTAN	FF	ARCTANGENT IN RADIANS OF Y/X
BESL	FS	BESSEL FUNCTION (GAMF)
COMP1	FF	EVALUATES REAL HYPERBOLIC TRIG FUNCTIONS
COMP2	FS	COMPLEX MULT. AND DIVISION
COMP3	FS	EVALUATES VARIOUS FUNCTIONS FOR COMPLEX ARGUMENT (COMP2)
ERRF	FF	ERROR FUNCTION
ERRINV	FF	INVERSE ERROR FUNCTION
FRESNL	FS	EVALUATES FRESNEL INTEGRALS
GAMF	FF	GAMMA FUNCTION
JACELF	FS	EVALUATES JACOBIAN ELLIPTIC FUNCTIONS SN, CN, DN
ORTHF	FF	EVALUATE ORTHOGONAL POLYNOMIALS
STIRLING	FP-S	N FACTORIAL BY STIRLINGS APPROXIMATION
TMFCEV	B	EVALUATE DAMPED OR UNDAMPED FOURIER SERIES

***LOGIC AND NUMBER THEORY*

**

4SQRS	B	WRITES INTEGERS AS SUM OF SQUARES OF FOUR INTEGERS
BASE	FP	CONVERTS NUMBERS FROM ONE BASE TO ANOTHER
CONCLUDE	9	DETERMINES LOGICAL CONCLUSIONS FROM PROPOSITIONAL LOGIC
GCDN	FS	G.C.D. OF N INTEGERS

*****ST--STATISTICS*****

CURVE FITTING AND REGRESSION

CFIT	FP	LEAST SQRS. POLY. WITH RESTRAINTS
CURFIT	B	FITS SIX DIFFERENT CURVES BY LEAST SQRS
F0RIR	FP	LEAST SQUARES ESTIMATE OF FINITE FOURIER SERIES MODEL
FOURIER	B	COEFF OF FOURIER SERIES TO APPROX A FUNCTION
LINEFIT	FS	LEAST SQRS LINE
LINREG	B	LST.SQRS. BY LINEAR, EXPONENTIAL, OR POWER FUNCTION
LSPCFP	FP	LEAST SQRS POLYNOMIAL FIT
LSQMM	FS	GENERALIZED POLY FIT BY LEAST SQRS OR MIN-MAX
MREG1	FP	MULTIPLE LINEAR REGRESSION
MULFIT	B	MULTIPLE LINEAR FIT WITH TRANSFORMATIONS
ORPOL	FP	LEAST SQRS FIT WITH ORTHOGONAL POLYS
POLFIT	B	LEAST SQRS POLYNOMIAL FIT
POLFT	FP	LEAST SQRS POLYNOMIAL FIT
SMLRP	FP	MULTIPLE LINEAR REGRESSION
STAT20	B	EFFRØYMSØN'S MULTIPLE LINEAR REGRESSION ALGORITHM
STAT21	B	COMPUTES MULTIPLE LINEAR REGRESSIONS

ANALYSIS OF VARIANCE

ANOVA	FP	ONE OR TWO WAY ANALYSIS OF VARIANCE
ANVA1	FP	ONEWAY ANALYSIS OF VARIANCE
ANVA3	FP	THREE WAY ANALYSIS OF VARIANCE
ANVA5	FP	MULTIPLE VARIANCE ANALYSIS
KRUWAL	FP	KRUSKAL-WALLIS 2-WAY VARIANCE (XINGAM)
ONEWAY	B	ONEWAY ANALYSIS OF VARIANCE
STAT13	B	ANALYSIS OF VARIANCE TABLE, 1-WAY RANDOM DESIGN
STAT14	B	ANALYSIS OF VARIANCE TABLE FOR RANDOMIZED BLOCK DESIGN
STAT15	B	ANALYSIS OF VARIANCE TABLE FOR SIMPLE LATIN-SQ DESIGN
STAT16	B	ANALYSIS OF VARIANCE TABLE, GRAECO-LATIN SQUARE DESIGN
STAT17	B	ANOVA TABLE OF BALANCED INCOMPLETE BLOCK DESIGN
STAT18	B	ANALYSIS OF VARIANCE TABLE, YÖUDEN SQUARE DESIGN
STAT33	B	ANALYSIS OF VARIANCE TABLE, 1-WAY RANDOM DESIGN

PROBABILITY DISTRIBUTIONS

ANPF	FF	NORMAL PROBABILITY FUNCTION (ERRF)
BETA	FF	BETA DISTRIBUTION
BINDIS	B	BINOMIAL PROBABILITIES
EXPLIM	B	EXPONENTIAL DISTRIBUTIONS
P0ISSON	FF	POISSON DISTRIBUTION FUNCTION
PR0BC	FP	PROBABILITIES OF COMBINATIONS OF RANDOM VARIABLES
PR0VAR	B	NORMAL AND T-DISTRIBUTION
TDIST	FF	T-DISTRIBUTION (BETA)
XINGAM	FF	INCOMPLETE GAMA FUNCTION

CONFIDENCE LIMITS

BAYES	B	DIFFERENCE OF MEANS IN NON-EQUAL VARIANCE
BIC0NF	B	C0NF. LIMITS FOR POPULATION PROPORTION (BINOMIAL)
BIN0M	FP	BINOMIAL PROBABILITIES AND CONFIDENCE BANDS
C0LINR	B	CONFIDENCE LIMITS ON LINEAR REGRESSIONS
C0NBIN	B	C0NF. LIMITS FOR POPULATION PROPORTION (NORMAL)
C0NDIF	B	DIFFERENCE OF MEANS IN EQUAL VARIANCE
C0NLM	B	C0NF. LIMITS FOR A SAMPLE MEAN
STAT05	B	CONFIDENCE INTERVAL FOR MEAN BY SIGN TEST
STAT06	B	CONFIDENCE LIMITS, WILCOXON SIGNED RANK SUM TEST

***HYPOTHESIS TESTING*

**

BITEST	B	TEST OF BINOMIAL PROPORTIONS
CHISQR	FS	CHI-SQUARE CALCULATIONS
C0RREL	FP	C0NTINGENCY C0EFFICIENT (XINGAM)
C0RRL2	FP	C0RRELATION C0EFFICIENT (TDIST,BETA)
K0K0	FP	K0LM0G0R0V-SMIRN0V TW0 SAMPLE TEST (XINGAM)
SEVPR0	B	CHI-SQUARE
STAT01	B	MEAN, STD OF MEAN, ... , T-RATIO, 2 GROUPS, PAIRED
STAT02	B	MEANS, VARIANCES, AND T-RATIO 2 GROUPS, UNPAIRED DATA
STAT04	B	CHI-SQUARE AND PROBABILITIES, 2X2 TABLES
STAT08	B	COMPARES TWO GROUPS OF DATA USING THE MEDIAN TEST
STAT09	B	COMPARE 2 DATA GROUPS, MANN-WHITNEY 2-SAMPLE RANK TEST
STAT11	B	SPEARMAN RANK C0RRELATION C0EF. FOR 2 SERIES OF DATA
STAT12	B	COMPUTES C0RRELATION MATRIX FOR N SERIES OF DATA
TAU	FP	KENDALL-RANK C0RRELATION

DESCRIPTIVE STATISTICS

MANDSD	B	FIND MEAN, VARIANCE, STD
STAT	FP	FIND SEVERAL STATISTICS FOR SAMPLE DATA (ANPF,ERRF)
STATAN	B	FIND VARIOUS STATISTICAL MEASURES
TESTUD	B	SAMPLE STATISTICS
UNISTA	B	DESCRIPTION OF UNI-VARIANT DATA

RANDOM NUMBER GENERATION

FLAT	GRF	UNIFORM RANDOM NUMBER GENERATOR
FLATS0RC	C	CARDIN SOURCE FILE FOR FLAT
RANDX	FF	RANDOM #'S, UNIFORM DIST. BETWEEN 0 AND 1
RNDNRM	FF	CALCULATES NORMAL RANDOM NUM. (FLAT)
UNIFM	GRF	UNIFORM RANDOM NUMBER GENERATOR
UNIFMS0R	C	CARDIN SOURCE FILE FOR UNIFM
URAN	GRF	UNIFORM RANDOM NUMBER GENERATOR
URANS0RC	C	CARDIN SOURCE FILE FOR URAN
XN0R1	FF	NORMAL RANDOM NUMBERS, VARIABLE MEAN, STD (RANDX)
XN0RM	FF	NORMAL RANDOM NUMBERS, MEAN 0, STD 1. (RANDX)

MISCELLANEOUS STATISTICS

FACTAN	FP	FACTOR ANALYSIS
STADES		EXPLANATION OF C0LINR, CURFIT, MULFIT, UNISTA

*****MS--MANAGEMENT SCIENCE AND OPTIMIZATION*

LINEAR PROGRAMMING

ASIGNIT B THE ASSIGNMENT PROBLEM
LINPRO B LINEAR PROGRAMMING
LNPROG FP LINEAR PROGRAMMING
SIMPLEX B LINEAR PROGRAMMING BY THE SIMPLEX METHOD
TRANSP0 B THE TRANSPORTATION PROBLEM
UNDEQ FS FINDS A SOLUTION FOR AN UNDERDETERMINED LINEAR SYSTEM

INTEGER PROGRAMMING

INTOI FP ZIANTS' MODIFICATION OF BALAS' ZERO-ONE ALGORITHM
INTLP FP GOMORY'S PURE AND MIXED INTEGER PROGRAMMING

NON-LINEAR OPTIMIZATION

CSM FS OPTIMIZE A LINEARLY CONSTRAINED CONVEX FUNCTION(UNDEQ)
DAVIDON B DAVIDON'S UNCONSTRAINED OPTIMIZATION
GEOSIM B HEURISTIC SCHEDULING OF N JOBS IN A M MACHINE SHOP
GPROG FHP SOLVES GEOMETRIC PROGRAMMING PROBLEMS
GPROG-S0 C CARDIN SOURCE FILE FOR GPROG (UNDEQ;CSM)
JSSIM B SCHEDULES N JOBS IN A SHOP WITH M MACHINES
LAYOUT B OPTIMIZES A PLANT LAYOUT ACCORDING TO VOLLMANN-RUML MODE
LOGIC3 FP UNCONSTRAINED OPTIMIZATION
MAXOPT FP UNCONSTRAINED OPTIMIZATION

NETWORK ANALYSIS

CPM FP CRITICAL PATH METHOD
CPML00P FP DETECTS AND LISTS LOOPS IN A CPM NETWORK
KILTER FP 'OUT OF KILTER' ALGORITHM (MINIMUM COST CIRCULATION)
MAXFLOW FP MAXIMUM FLOW THRU NETWORK
PERT B SIMPLE ANALYSIS OF A PERT NETWORK
SHORTEST FP SHORTEST PATH - MIN SPANNING TREE

FORECASTING

COEFS B DETERMINE SEASONAL COEFFICIENTS ON TWO CYCLES
COMBI B DETERMINES ECONOMIC ORDER QUANTITY FOR INVENTORY ITEMS
OPTIM F OPTIMUM SERVICE LEVEL FOR ONE INVENTORY ITEM
SMOOTH FS TRIPLE SMOOTHING OF A TIME SERIES
TCAST FHP TIME SERIES FORECASTING

SIMULATION

GASPDATA E DATA FILE FOR SAMPLE PROGRAM GASPSAMP
GASPIIA FS 'GASP' SIMULATION SYSTEM
GASPSAMP FP SAMPLE PROGRAM FOR GASPIIA (GASPIIA;GASPDATA)

*****EN--ENGINEERING*****

ACNET FP FREQUENCY RESPONSE OF A LINEAR CIRCUIT
BEMDES B STEEL BEAM SELECTION
GCVSIZ B GAS CONTROL VALVE COEFF.
LCVSIC B LIQUID CONTROL VALVE COEFF.
LFILTR B SYNTHESIZES ACTIVE LOW-PASS FILTERS (LFLDAT)
LFLDAT DATA FOR LFILTR
LFLTIN INSTRUCTIONS FOR LFILTR
LPFILT B DESIGN LOW PASS FILTERS
NLNET FP GENERAL STEADY-STATE CIRCUIT ANALYSIS
OTT0 B OTT0 CYCLE OF ENGINE
PAVEIT B CALCULATES \$ COST AND TONS OF MATERIAL TO PAVE A ROAD
PVT FP FINDS MOLAR VOLUME OF A GAS GIVEN TEMPERATURE AND PRES.
SCVSIZ B STEAM CONTROL VALVE COEFF.
SECAP B STEEL SECTION CAPACITIES

*****GP-GEOMETRIC AND PLOTTING*****

CIRCLE	B	DIVIDES A CIRCLE INTO N EQUAL PARTS
PL0T	FS	PLOTS UP TO 9 CURVES SIMULTANEOUSLY
PL0TT0	B	SIMULTANEOUSLY PLOTS 1 TO 6 FUNCTIONS
PL0T1	FS	PLOTS UP TO 9 CURVES SIMULTANEOUSLY
P0LPL0	FP	PLOTS EQNS IN POLAR COORDINATES
SPHERE	B	SOLVES ANY SPHERICAL TRIANGLE
TRIANG	B	SOLVES FOR ALL PARTS OF ANY TRIANGLE
TW0PL0	B	SIMULTANEOUSLY PLOTS 2 FUNCTIONS
XYPL0T	B	PLOTS SINGLE-VALUED FUNCTIONS

*****ED--EDUCATION AND TUTORIAL*****

DRIVES	FHP	DRIVER FOR EXPR, A COMPUTER ASSISTED INST. LANG.
EXPERN	E	EXPER TUTORIALS IN EXPR (N=1 TO 5) [PREPRS;DRIVES]
PREPRS	FHP	PREPROCESSOR FOR EXPR, A COMPUTER ASSISTED INST. LANG.

**

*****DE--DEMONSTRATION*****

AMAZE	B	CONSTRUCTS MAZES - EACH UNIQUE
BLKJAK	B	THE COMPUTER DEALS BLACKJACK
M00NER	B	SIMULATES A LUNAR LANDING(M00NER1,M00NER2)
M00NER1		DATA FILE FOR FOR M00NER
M00NER2		INSTRUCTIONS FILE FOR M00NER
P0PING	B	POPULATION PROJECTIONS FOR AN AREA
PRIME	B	PRIME FACTORIZATION OF A NUMBER
XMAS	B	A HOLIDAY SING-ALONG, CHRISTMAS CARD AND GREETINGS

*****UM--UTILITY AND MISCELLANEOUS*****

ADATER	FP-S	A CALENDER DATING ROUTINE
ACCESS	GS	GMAP SUBROUTINE TO USE T/S ACCESS SYSTEM (APPLIB
APARAM	GS	GMAP SUBROUTINE TO DETERMINE T/S OR BATCH MODE (APPLIB
APPLIB	GL	USERS LIBRARY OF FORTRAN CALLABLE GMAP ROUTINES(APPLIB-R
ASCB0D	GS	GMAP SUBROUTINE TO CONVERT ASCII TO BCD (APPLIB
BCDASC	GS	GMAP SUBROUTINE TO CONVERT BCD TO ASCII (APPLIB
CALLSS	GS	GMAP SUBROUTINE TO CALL A T/S SUBSYSTEM (APPLIB
CATALOG	E	CATALOG OF SERIES 6000/600 T/S LIBRARY (THIS FILE)
C0NVRT	B	CONVERTS MEASUREMENTS FROM ONE SCALE TO ANOTHER
DBLS0RT	FS	SORT TWO ARRAYS
DCS	FS	FORTRAN SUBR. TO TRANSFER CHARACTERS FROM STRING TO STRI
DEFIL	GS	GMAP SUBROUTINE TO CREATE TEMPORARY FILES (APPLIB
DESEQ	FP	STRIPS LINE SEQUENCE NUMBERS FROM A FILE
GMAP	FP	FORTRAN--AN INTERFACE TO GMAP ASSEMBLER (GMAP-S0R)
KIN	GS	GMAP SUBR. TO READ LAST LINE FROM TERM. IN BUFF. (APPLIB
REFORM	FP	REFORMATS A 'NF0RM' FORTRAN SOURCE FILE TO 'F0RM'
RLINE	FS	READS LINE, OPTIONALLY STRIPS LINE # & COUNTS ENTRIES
SGLS0RT	FS	SORT AN ARRAY
TLU1	FS	TABLE SEARCH
TPLS0RT	FS	SORT THREE ARRAYS
UAT0LA	GS	GMAP SUBR. TO CHANGE CASE OF ASCII CHAR. STRING (APPLIB

*****BF--BUSINESS AND FINANCE*****

ACCRUIT	B	COMPUTES AND PRINTS ACCRUED INTEREST ON INSTALLMENT LOAN
ANNUIT	B	ANNUITIES, LOANS, MORTGAGES
BALANCE	B	A PROGRAM TO RECONCILE A BANK STATEMENT BALANCE
BLDGCCST	B	ANALYZE BUILDING COSTS
BONDATA	B	ANALYSIS OF A BOND INVESTMENT PORTFOLIO
BONDPR	B	COMPUTES PRICE AND ACCRUED INTEREST OF A BOND
BONDSW	B	CALCULATES THE EFFECT OF A BOND SWITCH
BONDYD	B	COMPUTES BOND YIELDS
CASHFLOW	B	PREDICTS NEXT YEARS CASH FLOW
DEPREC	B	CALCULATES DEPRECIATION BY FOUR METHODS
INSTLO	B	CALCULATES MONTHLY PAYMENT SCHEDULE ON INSTALLMENT LOAN
INTRSTZ	B	INTEREST RATES REGARDLESS OF PAYMENT STREAM --REGULATION
INVANL	FP	RETURN ON INVESTMENT ANALYSIS
LESSEE	B	COMPARES A LEASE WITH PURCHASE OF EQUIPMENT
LESSIM	B	SIMULATES LESSOR'S CASH FLOW AND RATE OF RETURN
LESSOR	B	CALCULATES THE LESSORS CASH FLOW & RATE OF RETURN
MAKE-BUY	B	TO MAKE OR TO BUY DECISIONS
MGSIM	FP	SIMULATES COMPETITIVE INTERACTION OF COMPANIES
MGSIM-IN		ON LINE INSTRUCTIONS FOR MGSIM
MORTCST	B	MORTGAGE SCHEDULE FOR VARIOUS TERMS
MORTGAGE	FP	CALCULATES A MORTGAGE REPAYMENT SCHEDULE
NOM-EFF	B	COMPUTES MULTIPLE EFFECTIVE ANNUAL RATES OF INTEREST
RETURN	B	COMPUTES ANNUAL RETURNS FOR A SECURITY FROM ANNUAL DATA
RISKIT	B	RISK ANALYSIS BASED ON HERTZ'S SIMULATION MODEL
SALDATA	B	COMPUTES PROFITABILITY OF DEPARTMENTS OF A FIRM
SAVING	B	SAVINGS PLAN CALCULATIONS
SIMFUND	B	SIMULATES LONG-RUN PERFORMANCE OF FUNDS(SIMPL0T)
SIMPL0T	B	PL0TTING PROGRAM FOR SIMFUND HIST0GRAMS
SMLBUS	B	PAYMENT SCHEDULES FOR A SMALL BUSINESS ADMST. LOAN
TRUINT	B	INTEREST RATE CALCULATIONS
VALSTK	B	CALCULATES INTRINSIC VALUE OF STOCK--MOL0DOVSKY METHOD

END OF CATALOG

This FORTRAN program computes the earliest (ES) and latest (LS) permissible start times, earliest (EF) and latest (LF) finish times, total float (TF), and free float (FF) for the activities of a project network model. The program also determines the critical path through the network and, at the user's option, calculates direct cost, percent completion, and calendar dates.

RESTRICTIONS

1. The nodes (events) of the arrow diagram can be numbered randomly with any number from 0 to 4095.
2. Each activity must have a nonnegative duration expressed in whole days.
3. The maximum problem size is limited by:

$$2 * (\text{highest numbered node}) + (\# \text{ activities}) + 2 \leq MX$$

and, if calendar dating is desired:

$$(\text{project duration}) + 2 * (\text{highest numbered node}) + 3 \leq MX,$$

where MX is currently set to 3000. MX can be reset by changing the parameter statement near the beginning of the program.

NOTE: The head of the job arrow (J) does not have to be numbered larger than the tail (I). There can be more than one arrow with the same I and J, and the activities can be input in any order.

DATA FILE FORMAT

The data should be entered in a file before executing the program, either with or without line numbers. Three types of information must be entered in the data file: data file format information, activity data and, if desired, calendar dating information.

The first line in the file contains alphanumeric problem identification. If the first character of this line is numeric, the program assumes the data file was built with line numbers.

The second line gives the format of the activity data and is in the following format:

(line #)	$\left\{ \begin{array}{c} \text{COST} \\ \text{NCOST} \end{array} \right\}$	$\left\{ \begin{array}{c} \text{PRCNT} \\ \text{NPRCNT} \end{array} \right\}$	$\left\{ \begin{array}{c} \text{DESCR} \\ \text{NDESCR} \\ \text{LDESCR} \end{array} \right\}$	linesize, media
----------	---	---	--	-----------------

The entries on this line have the following meanings.

COST	Activity cost is included and is to be processed.						
NCOST	Activity cost is not included or is included but neither cost nor percent are to be processed.						
PRCNT	Activity percent completed is included and is to be processed.						
NPRCNT	Activity percent completed is not to be processed.						
DESCR	Alphanumeric activity identification is on the same line as the rest of the activity data and is to be processed. The identification must begin with a nonnumeric character.						
NDESCR	Alphanumeric activity identification is not to be processed but can still be included on the same line as the rest of the activity data.						
LDESCR	Alphanumeric activity identification is on the line following the rest of the activity data and is to be processed. The identification must begin with a nonnumeric character.						
linesize	The number of character positions per line on the output device.						
media	The media code to use in writing the network report data includes: <table data-bbox="475 891 1061 983"> <tr> <td>media = 3</td><td>BCD print line</td></tr> <tr> <td>media = 5</td><td>TS ASCII</td></tr> <tr> <td>media = 6</td><td>System standard ASCII</td></tr> </table>	media = 3	BCD print line	media = 5	TS ASCII	media = 6	System standard ASCII
media = 3	BCD print line						
media = 5	TS ASCII						
media = 6	System standard ASCII						

Beginning on the third line, the activity data is entered in free format as indicated below and as described in the first two lines of the data file. Note that all numeric values are entered in integer mode only. The items in parentheses below are optional:

(line #) I J duration (cost) (percent) (identification)

or

(line #) I J duration (cost) (percent)

(line #) identification

If calendar dating is desired, the last line should have I = J = duration = cost = percent = 0.

The program tests for errors and issues the following error messages:

1. I-J errors — I or J greater than 4095, negative, or I = J.
2. Problem too big for allocated storage.
3. Multiple start or finish nodes.
4. A loop in the network. In this case, the activity identified is either on the loop or on a sequence of jobs that pass through a node of the loop.

For calendar dating, the first line following the activities is the starting date in the following format. If the starting month number is negative, the calendar dating is ignored.

(line #) month #, day of month, day of week #

Following the starting date, the nonworking days of the week and the holidays are entered for each year the project is expected to cover. This data is entered one item per line in the following order:

```
(line #) last two digits of year
(line #) a nonworking day of week
      etc.
(line #) -1 (end of nonworking days)
(line #) holiday month #, day
      etc.
(line #) -1, -1 (end of holidays)
(line #) last two digits of next year
      etc.
      .
      .
      .
(line #) last holiday of last year
```

INSTRUCTIONS

If the data file was named 05 or has already been accessed under the alternate name 05, the program can be executed by typing:

```
RUN LIBRARY/CPM,R=(CORE=23)#05
```

If the data file has a name other than 05, for example, CPMDATA, the program can be executed by typing:

```
RUN LIBRARY/CPM,R=(CORE=23)#CPMDATA "05"
```

If the activity data is to be written to a file rather than to a terminal, type the following command line (depending upon particular core allocated):

```
RUN LIBRARY/CPM,R=(CORE=23)#05;06
```

The activity data will be written to file 06. Error messages and project summaries will still be sent to the terminal.

This program can also be executed in the batch environment by submitting a job similar to the following:

```
$ IDENT
$ OPTION          FORTRAN
```

CPM-4

```

$   FORTY          ASCII, LNO, NFORM, OPTZ
$   LIMITS         , 32K
$   PRMFL          S*, READ, SEQ, LIBRARY/CPM
$   EXECUTE
$   LIMITS         , 13K
$   PRMFL          05, READ, SEQ, userid/datafile
$   ENDJOB
  
```

SAMPLE PROBLEM

Figure 1 is an arrow diagram for a sample project. The problem is first solved without calendar dating and then with calendar dating.

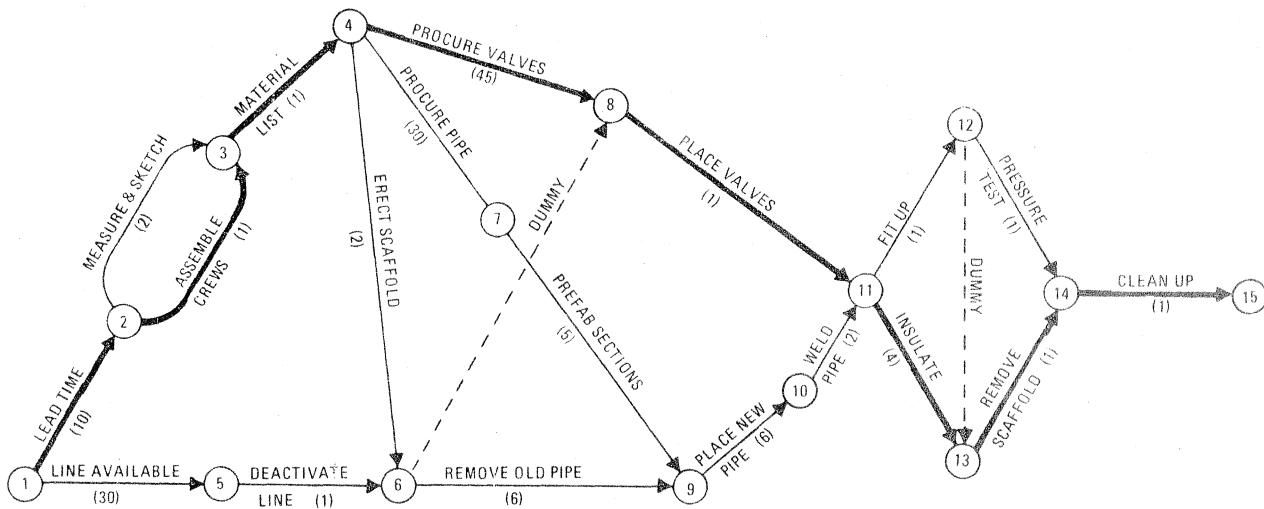


Figure 1. Sample Arrow Diagram for Renewal of Pipeline

SAMPLE SOLUTION 1 - WITHOUT CALENDAR DATING

*OLD 05
 READY
 *LIST

010 TEST CPM PROGRAM -- NO CALENDAR DATING
 020 COST PRCNT DESCR 118 5
 30 1 2 10 0 100
 040 1 5 30 0 100 LINE AVAILABLE
 050 2 3 2 25 100 MEASURE & SKETCH
 060 2 3 1 300 100 ASSEMBLE CREWS
 070 3 4 1 10 100 MATERIAL LIST
 080 5 6 1 75 50 DEACTIVATE LINE
 090 4 6 2 100 45 ERECT SCAFFOLD
 100 4 7 30 100 10 PROCURE PIPE
 110 4 8 45 150 15 PROCURE VALVES
 120 6 8 0 0 0 DUMMY
 130 7 9 5 220 0 PREFAB SECTIONS
 140 9 10 6 120 0 PLACE NEW PIPE
 150 10 11 2 150 0 WELD PIPE
 160 8 11 1 75 0 PLACE VALVES
 170 11 12 1 30 0 FIT UP
 180 11 13 4 80 0 INSULATE
 190 12 13 0 0 0 DUMMY
 200 12 14 1 20 0 PRESURE TEST
 210 13 14 1 20 0 REMOVE SCAFFOLD
 220 14 15 1 20 0 CLEAN UP

READY

*RUN LIBRARY/CPM,R=(CORE=23)#05

*** C P M ***

10 TEST CPM PROGRAM WITH CALENDAR DATING

END OF DATA AFTER	21 ACTIVITIES		14		15J					
I	J	DUR	COST	%	ES	EF	LS	LF	TF	FF
*	1	2	10	0 100	0	10	0	10	0	0
	1	5	30	0 100	0	30	27	57	27	0 LINE AVAILABE
*	2	3	2	25 100	10	12	10	12	0	0 MEASURE & SKETCH
	2	3	1	300 100	10	11	11	12	1	1 ASSEMBLE CREWS
*	3	4	1	10 100	12	13	12	13	0	0 MATERIAL LIST
	5	6	1	75 50	30	31	57	58	27	0 DEACTIVATE LINE
	4	6	2	100 45	13	15	56	58	43	16 ERECT SCAFFOLD
	4	7	30	100 10	13	43	16	46	3	0 PROCURE PIPE
*	4	8	45	150 15	13	58	13	58	0	0 PROCURE VALVES
	6	8	0	0 0	31	31	58	58	27	27 DUMMY
	7	9	5	220 0	43	48	46	51	3	0 PREFAB SECTIONS
	9	10	6	120 0	48	54	51	57	3	0 PLACE NEW PIPE
	10	11	2	150 0	54	56	57	59	3	3 WELD PIPE
*	8	11	1	75 0	58	59	58	59	0	0 PLACE VALVES
	11	12	1	30 0	59	60	62	63	3	0 FIT UP
*	11	13	4	80 0	59	63	59	63	0	0 INSULATE
	12	13	0	0 0	60	60	63	63	3	3 DUMMY
	12	14	1	20 0	60	61	63	64	3	3 PRESURE TEST
*	13	14	1	20 0	63	64	63	64	0	0 REMOVE SCAFFOLD
*	14	15	1	20 0	64	65	64	65	0	0 CLEAN UP

TOTAL PROJECT COST = 1495

65 WORKING DAYS REQUIRED TO COMPLETE PROJECT

38% OF TOTAL PROJECT COMPLETED

30% OF CRITICAL PATH ACTIVITIES COMPLETED

CPM-6

SAMPLE SOLUTION 2 -- WITH CALENDAR DATING

*OLD 05

READY

*LIST

```
010 TEST CPM PROGRAM WITH CALENDER DATING
020 COST PRGNT DESCR 118 5
30 1 2 10 0 100
040 1 5 30 0 100 LINE AVAILABLE
050 2 3 2 25 100 MEASURE & SKETCH
060 2 3 1 300 100 ASSEMBLE CREWS
070 3 4 1 10 100 MATERIAL LIST
080 5 6 1 75 50 DEACTIVATE LINE
090 4 6 2 100 45 ERECT SCAFFOLD
100 4 7 30 100 10 PROCURE PIPE
110 4 8 45 150 15 PROCURE VALVES
120 6 8 0 0 0 DUMMY
130 7 9 5 220 0 PREFAB SECTIONS
140 9 10 6 120 0 PLACE NEW PIPE
150 10 11 2 150 0 WELD PIPE
160 8 11 1 75 0 PLACE VALVES
170 11 12 1 30 0 FIT UP
180 11 13 4 80 0 INSULATE
190 12 13 0 0 0 DUMMY
200 12 14 1 20 0 PRESURE TEST
210 13 14 1 20 0 REMOVE SCAFFOLD
220 14 15 1 20 0 CLEAN UP
230 0 0 0 0 0 END OF ACTIVITY DATA
240 12 29 2 STARTING MONTH, DAY, DAY OF WEEK
250 69 STARTING YEAR, WORK SCHEDULE FOLLOWS
260 1 IS A NON-WORK-DAY OF WEEK
270 7 DITT0
280 -1 END OF NON-WORK-DAYS
290 12 25 CHRISTMAS IS A HOLIDAY
300 -1 -1 END OF HOLIDAYS FOR THIS YEAR
310 70 NEXT YEAR
320 1 IS A NON-WORK-DAY
330 7 DITT0
340 -1 END OF NON-WORK-DAYS
350 1 1 NEW YEARS IS A HOLIDAY
360 2 12 LINCOLN'S BIRTHDAY
370 3 27 GOOD FRIDAY
```

READY

*RUN LIBRARY/CPM,R=(CORE=23)#05

*** C P M ***

10 TEST CPM PROGRAM WITH CALENDER DATING

	I	J	DUR	COST	%	ES	EF	LS	LF	TF	FF
*	1	2	10	0	100	12/29/69	1/13/70	12/29/69	1/13/70	0	0
	1	5	30	0	100	12/29/69	2/10/70	2/ 5/70	3/20/70	27	0
*	2	3	2	25	100	1/13/70	1/15/70	1/13/70	1/15/70	0	0
	2	3	1	300	100	1/13/70	1/14/70	1/14/70	1/15/70	1	1
*	3	4	1	10	100	1/15/70	1/16/70	1/15/70	1/16/70	0	0
	5	6	1	75	50	2/10/70	2/11/70	3/20/70	3/23/70	27	0
	4	6	2	100	45	1/16/70	1/20/70	3/19/70	3/23/70	43	16
	4	7	30	100	10	1/16/70	3/ 2/70	1/21/70	3/ 5/70	3	0
*	4	8	45	150	15	1/16/70	3/23/70	1/16/70	3/23/70	0	0
	6	8	0	0	0	2/11/70	2/11/70	3/23/70	3/23/70	27	27
	7	9	5	220	0	3/ 2/70	3/ 9/70	3/ 5/70	3/12/70	3	0
	9	10	6	120	0	3/ 9/70	3/17/70	3/12/70	3/20/70	3	0
	10	11	2	150	0	3/17/70	3/19/70	3/20/70	3/24/70	3	3
*	8	11	1	75	0	3/23/70	3/24/70	3/23/70	3/24/70	0	0
	11	12	1	30	0	3/24/70	3/25/70	3/30/70	3/31/70	3	0
*	11	13	4	80	0	3/24/70	3/31/70	3/24/70	3/31/70	0	0
	12	13	0	0	0	3/25/70	3/25/70	3/31/70	3/31/70	3	3
	12	14	1	20	0	3/25/70	3/26/70	3/31/70	4/ 1/70	3	3
*	13	14	1	20	0	3/31/70	4/ 1/70	3/31/70	4/ 1/70	0	0
*	14	15	1	20	0	4/ 1/70	4/ 2/70	4/ 1/70	4/ 2/70	0	0

TOTAL PROJECT COST = 1495

65 WORKING DAYS REQUIRED TO COMPLETE PROJECT

PROJECT START DATE 12/29/69

PROJECT COMPLETION DATE 4/ 2/70

38% OF TOTAL PROJECT COMPLETED

30% OF CRITICAL PATH ACTIVITIES COMPLETED

This FORTRAN program identifies multiple loops in a CPM-type network. It also isolates other nonlooping elements, such as parallel legs (alternate paths in a loop) and interconnecting strings.

METHOD

This program uses the algorithm described by J. A. Chance.¹

INSTRUCTIONS

Enter the network description in a data file according to the following rules:

1. The first line contains any alphanumeric problem identification. The routine prints this line on the output report. It also determines whether the file contains preceding line numbers based on the first character of this line (i.e., if the first character of the first line of the data file is numeric, it is assumed the entire file was built with line numbers).
2. The second line informs the program if an alphanumeric activity description has been entered on the line following each activity description. If this line contains an 'L' in any position, it is assumed the data file has been built using two lines for each activity. If it does not contain an 'L', each activity description is contained on only one line.
3. The following lines contain the activity descriptions in the order
(line #) from node, to node (description)
or, if the second line contains an 'L',
(line #) from node, to node (description)
(line #) description
4. Optionally, the end of data can be flagged by an activity for which
from node = to node = 0

NOTE: The data file format for the program CPM is compatible with the data file requirements for this program.

The routine may fail to detect all of the loops in a network by incorrectly classifying some as parallel legs. A second pass is made through the network to attempt to identify any loops that might have been missed on the first pass. However, some parallel legs and interconnecting strings may then be lost.

¹Chance, J. A., "Multiple Loop Detection in Network Models," General Electric Technical Information Series, Publication No. R64CD1, 1964.

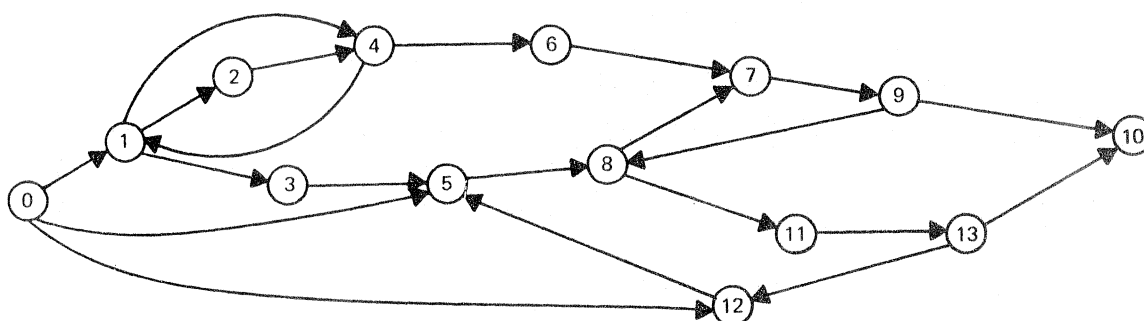
RESTRICTIONS

$$(\text{MAX. NODE \#}) + (\text{MAX. \# OF ACTIVITIES}) * 3 < \text{MAXTAB}$$

Also the node numbers must satisfy

$$0 \leq \text{node \#} < 16384$$

Isolate the loops in the following network.



```
*NEW
READY
*010 TEST DATA FOR CPML00P
*020 NC0ST NPRCNT NDESCR 72 5
*030 0 1
*040 0 5
*050 0 12
*060 1 2
*070 1 3
*080 1 4
*090 2 4
*100 3 5
*110 4 1
*120 4 6
*130 5 8
*140 6 7
*150 7 9
*160 8 7
```


*170 8 11
 *180 9 8
 *190 9 10
 *200 11 13
 *210 12 5

*220 13 10
 *230 13 12
 *SAVE LOOPDATA
 DATA SAVED--LOOPDATA

*LIB CPML00P
 READY
 *RUN
 ENTER NAME OF DATA FILE
 *LOOPDATA
 010 TEST DATA FOR CPML00P

SIMPLE LOOP(S)

7 9 8 7 9 8

 1 4 4 1

 5 8 8 11 11 13 12 5 13 12

The file GASPIIA contains a set of FORTRAN subroutines that provide the programmer with a FORTRAN-based simulation language that simulates event-oriented systems. GASPSAMP and GASPDATA are sample problem files.

METHOD

GASPIIA provides routines for event control, statistical collection and computation, report generation, and random number generation. The user must supply the main program, the event routines, and an optional output routine.

NOTE: GASPIIA calls a uniform random number generator in the form:

Y = UNIFM1(X). The library programs FLAT, URAN, or UNIFM can be used.

INSTRUCTIONS

Log onto time sharing under the FORTRAN subsystem, and write a main program and subprograms as needed for the system to be simulated. Then give the command:

RUN*; GASPIIA; FLAT

The program will type:

NAME OF INPUT FILE

Respond with the name of a previously prepared input file.

Input performed by the GASPIIA subroutines will be from this file. If blanks are entered for a file name, the file is the terminal device. The input file is in free-field format except for the first line, which is formatted as described in SIMULATION WITH GASPII (see References). The input file should not have line numbers. The GASP storage map may be listed.

REFERENCES

Pritsker, Alan A., Kiviat, Phillip J., Simulation with GASPII, Englewood Cliffs, New Jersey, Prentice Hall, 1969.

SAMPLE PROBLEM

Sample problem 5 from SIMULATION WITH GASPII has been programmed. The user-supplied coding is stored in the file GASPSAMP. Problem data is stored in the file GASPDATA.

GASPIIA-2

SAMPLE SOLUTION

The execution of GASPIIA is demonstrated by running the sample program with GASPSAMP and GASPDATA. The GASPDATA file is also listed.

*LIST GASPDATA

```
PRITSKER A      504151968    1
 3 2 2 4 20 1 4 22 4
20 20
 1 2 3 2
 1 1 1 1
.4 0.0 10.0 1.
.25 0.0 10.0 1.
.5 0.0 10.0 1.
0 1 0 0 0.0 400. 567
-1 0
 1 1
0.1 0.0 0.1 0.0
 1 2
1.0 0.0 0.0 0.0
 1 3
1.0 0.0 0.0 0.0
 2 0
 0.0 0 0 0
 2 0
 0.0 0 0 0
 2 0
 0 0 0 0
 1 4
300. 0 0 0
 0 0
```

READY

*OLD GASPSAMP

READY

*RUN *; GASPIIA; FLAT
NAME OF INPUT FILES?
= GASPDATA

SIMULATION PROJECT NO. 5 BY PRITSKER A

DATE 4/ 15/ 1968 RUN NUMBER 1

PARAMETER NO.	1	0.4000	0.	10.0000	1.0000
PARAMETER NO.	2	0.2500	0.	10.0000	1.0000
PARAMETER NO.	3	0.5000	0.	10.0000	1.0000

DO YOU WANT TO SEE A GASP JOB STORAGE DUMP?

0=NO, 1=YES

0

***INTERMEDIATE RESULTS**

GASP SUMMARY REPORT

SIMULATION PROJECT NO. 5 BY PRITSKER A

DATE 4/ 15/ 1968 RUN NUMBER 1

PARAMETER NO.	1	0.4000	0.	10.0000	1.0000
PARAMETER NO.	2	0.2500	0.	10.0000	1.0000
PARAMETER NO.	3	0.5000	0.	10.0000	1.0000

**

GENERATED DATA**

CODE	MEAN	S.DEV.	MIN.	MAX.	OBS.
1	3.2337	1.5581	0.2531	7.8550	532
2	0.5693	0.5704	0.0035	3.4906	532

TIME GENERATED DATA

CODE	MEAN	STD.DEV.	MIN.	MAX.	TOTAL TIME
1	5.6803	2.0974	0.	8.0000	302.8572
2	0.4330	0.4955	0.	1.0000	302.8572
3	0.9108	0.2850	0.	1.0000	302.8572
4	47.0627	49.9136	0.	100.0000	302.8572

GENERATED FREQUENCY DISTRIBUTIONS

CODE	HISTOGRAMS											
1	10	18	32	51	74	80	73	53	32	32	20	
	22	14	12	7	2	0	0	0	0	0	0	
2	0	140	111	72	48	41	35	18	21	13	7	
	6	5	6	3	2	0	0	1	1	2	0	

FILE PRINTOUT, FILE NO. 1

AVERAGE NUMBER IN FILE WAS,	2.3528
STD. DEV	0.5758
MAXIMUM	4

FILE CONTENTS

NSET

THE FILE IS EMPTY

FILE PRINTOUT, FILE NO. 2

AVERAGE NUMBER IN FILE WAS,	2.2755
STD. DEV	1.4822
MAXIMUM	4

GASPIIA-4

FILE CONTENTS

NSET

THE FILE IS EMPTY

FILE PRINTOUT, FILE NO. 3

AVERAGE NUMBER IN FILE WAS,	1.5625
STD. DEV	0.7314
MAXIMUM	2

FILE CONTENTS

NSET

THE FILE IS EMPTY

FILE PRINTOUT, FILE NO. 4

AVERAGE NUMBER IN FILE WAS,	0.4706
STD. DEV	0.4991
MAXIMUM	1

FILE CONTENTS

NSET

THE FILE IS EMPTY

MEAN TIME BETWEEN ARRIVALS = 0.40
MEAN SERVICE TIME FOR STATION 1 = 0.25
MEAN SERVICE TIME FOR STATION 2 = 0.50
PERCENT OF ITEMS SUBCONTRACTED = 27.21
NUMBER OF ITEMS SUBCONTRACTED = 197.
TOTAL ITEMS = 724.

This FORTRAN program solves the zero-one integer programming problem using a modification of Balas' method of implicit enumeration.

INSTRUCTIONS

To use this program, formulate the problem to be solved according to the following standard:

Minimize

$$\sum_{i=1}^n a_{oi} x_i$$

subject to

$$a_{jo} \geq \sum_{i=1}^n a_{ji} x_i \quad j=1, \dots, m$$

and $x_i = 0$ or 1 , $i = 1, \dots, n$

Hence, n is the number of variables and m is the number of constraints. If m is greater than 11 or n is greater than 28, the dimensions in the program must be increased.

All of the coefficients, a_{ij} , should be integer. Establish a data file with line numbers in the following format:

line #, m, n

line #, 0, $a_{o1}, \dots, a_{on}$

line #, $a_{10}, a_{11}, \dots, a_{17}$

.

.

.

line #, $a_{m0}, a_{m1}, \dots, a_{mn}$

Note that a_{i0} for $i = 1$ to m are the right-hand side constants. The first entry after the line number on the second line is ignored by this program. It is included to maintain consistency with the data file format used by the INTLP program.

The algorithm asks for a maximum number of iterations to be performed. If the optimum is not found and confirmed within the limit given, data is written on a temporary file 20. The user can then give a new limit on the number of iterations or can stop the

program by giving 0 as the new limit. If the file 20 is saved, the user can continue the problem at a later time, at the point where he left off.

On execution, the program asks if the problem is a new or a restart using data on the file START. It also requests the name of the original data file. If the problem is a new one, the user can specify some variables to be set to one. The algorithm will never consider the cases with those variables set to 0 (see Note 2). If no variables are to be set to 1, enter 0 for the first index.

The program prints out the first feasible answer found and all subsequent feasible answers whose objective value is better than the last feasible answer.

- NOTES:
1. Although the optimum answer is usually easily found, it then requires many iterations to verify that it is optimum.
 2. The option of setting variables to one may reduce the number of iterations needed to find the optimum; however, this optimum may not be the optimum to the original problem.

REFERENCES

Zionts, Stanley, Implicit Enumeration Using Bonds on Variables: A Generalization of Balas' Additive Algorithm for Solving Linear Programs With Zero-One Variables, presented at the Operational Research Society of India Annual Meeting, Calcutta, India, November 1968.

SAMPLE PROBLEM

Solve the problem, with four constraints and 20 variables, which has as the coefficient vector of the objective function to be minimized

(3, 2, 5, 8, 6, 9, 11, 4, 5, 6, 11, 2, 8, 5, 8, 7, 3, 9, 2, 4)

The coefficient matrix of the constraints:

$$\begin{bmatrix} -6 & 5 & -8 & -3 & 0 & -1 & -3 & -8 & -9 & 3 & -8 & 6 & -3 & -8 & -6 & 7 & 6 & -2 & 3 & -7 \\ -1 & 3 & 3 & 4 & 1 & 0 & 4 & -1 & -6 & 0 & 8 & 0 & 1 & -5 & -4 & -1 & -9 & -7 & 2 & 2 \\ 3 & 6 & 1 & -3 & -5 & 6 & -9 & 6 & 3 & -9 & -6 & -3 & -6 & -6 & 6 & 2 & -7 & -6 & 0 & -7 \\ 1 & 4 & 2 & -1 & 1 & 1 & 1 & -7 & -8 & -9 & -8 & -7 & -9 & 1 & 1 & 0 & 1 & 2 & 1 & -3 \end{bmatrix}$$

and the right hand sides of the constraints

(-25, -13, -15, -13)

SAMPLE SOLUTION

The data was stored in the file IN01IN. The program ran with a maximum of 100 iterations, and then stopped. The restart capability was illustrated by running the program again with a maximum of 500 iterations. No new feasible solutions were found, so the maximum was increased to 900 iterations. Notice that when signing off, the user is given the option of saving the temporary file 20.

*RUN

INT01

O=NEW PROBLEM, 1=RESTART ON OLD PROBLEM
= 0

NAME OF THE DATA FILE

= IN01IN

MAX # OF ITER BEFORE DECISION POINT

= 100

INDICES OF VARIABLES TO BE SET 1, ONE AT A TIME
(0 STOPS SCAN)

= 0

FEASIBLE POINT

ALL VARIABLES ARE 0 EXCEPT:

X(9)=1

X(13)=1

X(14)=1

X(18)=1

X(20)=1

Z= 31 ITER= 5

FEASIBLE POINT

ALL VARIABLES ARE 0 EXCEPT:

X(1)=1

X(9)=1

X(13)=1

X(14)=1

X(17)=1

X(20)=1

Z= 28 ITER= 13

FEASIBLE POINT

ALL VARIABLES ARE 0 EXCEPT:

X(5)=1

X(8)=1

X(9)=1

X(14)=1

X(17)=1

X(20)=1

Z= 27 ITER= 83

AT ITER 100 RESTART FILE BUILT

WHAT IS NEW ITMAX (0=STOP)

= 0

PROGRAM STOP AT 700

*RUN

INT01

O=NEW PROBLEM, 1=RESTART ON OLD PROBLEM

= 1

NAME OF THE DATA FILE

= IN01IN

MAX # OF ITER BEFORE DECISION POINT

500

INT01-4

BEST SOLUTION FOUND PREVIOUSLY

ALL VARIABLES ARE 0 EXCEPT:

X(5)=1

X(8)=1

X(9)=1

X(14)=1

X(17)=1

X(20)=1

Z= 27 ITER= 100

AT ITER 500 RESTART FILE BUILT

WHAT IS NEW ITMAX (0=STOP)

= 900

OPTIMUM

ALL VARIABLES ARE 0 EXCEPT:

X(5)=1

X(8)=1

X(9)=1

X(14)=1

X(17)=1

X(20)=1

Z= 27 ITER= 890

*BYE

1 TEMPORARY FILES CREATED.

20 ?

The data was saved in the file IN01IN and is listed below:

*LIST IN01IN

010 4,20

020 0, 3,2,5,8,6,9,11,4,5,6,11,2,8,5,8,7,3,9,2,4

030 -25, -6,5,-8,-3,0,-1,-3,-8,-9,3,-8,6,-3,-8,-6,7,6,-2,3,-7

040 -13, -1,3,3,4,1,0,4,-1,-6,0,8,0,1,-5,-4,-1,-9,-7,2,2

050 -15, 3,6,1,-3,-5,6,-9,6,3,-9,-6,-3,-6,-6,6,2,-7,-6,0,-7

060 -13, 1,4,2,-1,1,1,1,-7,-8,-9,-8,-7,-9,1,1,0,1,2,1,-3

READY

*

This FORTRAN program uses Gomory's method to solve both the pure-integer and mixed-integer programming problems. The user can actively interface with the program by changing the method of picking the new constraints and by being able to add certain information to the problem in an effort to speed up convergence.

INSTRUCTIONS

To use this program, formulate the problem to be solved according to the following standard:

Minimize

$$a_{00} + \sum_{i=1}^n a_{0i} x_i$$

subject to

$$a_{j0} \geq \sum_{i=1}^n a_{ji} x_i, \quad j=1, \dots, m$$

Hence n is the number of variables and m is the number of constraints. All of the coefficients a_{ij} should be integer.

Establish a data file with line numbers in the following format:

line # , m , n

line # , a_{00} , a_{01} , $\dots$, a_{0n}

line # , a_{m0} , a_{m1} , $\dots$, a_{mn}

On executing the program, the user is asked to supply the method to be used; the changes, if any, to be made to the data; and when to print out the iteration log. Every five times the log is printed out, the user can decide if he wants to stop, or continue the solution, or restart the problem from the beginning. If he chooses to continue, he can change the printing of the iteration log and the method to be used. The following items should be supplied when requested by the program.

- The name of the data file
- LSTART — The iteration when the log should be first printed out
- LOFTEN — How often should the log be printed

- LMUCH — How much of the log does the user wish to see
 - LMUCH = 1) Print the entire log
 - 2) Do not print the values of the variables
- METHOD — The method to be used in choosing the new constraints

Methods 1 through 5 are for the pure integer case. Method 6 is the mixed case.

METHOD = 1 Generate the new constraint from the row with the greatest RHS fractional part.

2 Generate the new constraint from the row with the smallest average fractional part of the coefficients.

3 Generate the new constraint from the row with the smallest ratio of the RHS fractional part and the average fractional part of the coefficients

4 Use the Euclidian algorithm to generate the new constraint from the linear combination of two rows that are likely to produce a deep cut in the axis along which the objective function decreases most slowly.

5 Use the rows cyclicly to generate the new constraints

6 This method is for the mixed case. Its selection causes the program to ask for the number of variables constrained to integer values and their indices. Care should be used with this method to avoid changing the problem by specifying different variables to be integer at the various decision points.
- CHANGES — The number of changes to be made to the data before starting. If there are changes to be made, the user will be asked

VARIABLE #, VALUE

for each change. The permissible responses are:

 - (a) A variable number (1 through N) and the value at which it is fixed. This option eliminates the variable and its column from any manipulation by the program.
 - (b) 0, X. This option appends the constraint: objective value $\geq X$ to the problem.
 - (c) -1, X. This option appends the constraint: objective value $\leq X$ to the problem.
- NEXT — After printing the iteration log five times, the user is given the choice of
 - 1) Stop
 - NEXT = 2) Continue
 - 3) Begin the problem again

NOTE: If the objective function has not changed for several iterations, the program may be looping indefinitely through the same points. If this occurs, another method should be tried. Also, another method could be tried profitably whenever the change in the objective function becomes small.

RESTRICTIONS

The program currently cannot handle more than 14 variables and approximately 16 constraints. Also, Gomory's method is known to converge slowly, especially for large problems. However, by wisely using the interface capabilities of this program, many problems can be solved with reasonable effort.

REFERENCES

Gomory, R.E., "An Algorithm for Integer Solutions to Linear Programs," in Recent Advances in Optimization Theory, Ed: Graves & Wolfe, McGraw-Hill, 1963.

Trauth & Woosley, Mesa, An Heuristic Integer Programming Technique, Sandia Laboratories, Albuquerque, New Mexico.

SAMPLE PROBLEM

Minimize $-x_3 -x_4 -x_5$

subject to:

$$180 \geq 20 x_1 + 30 x_2 + x_3 + 2 x_4 + 2 x_5$$

$$150 \geq 30 x_1 + 20 x_2 + 2 x_3 + x_4 + 2 x_5$$

$$0 \geq -60 x_1 + x_3$$

$$0 \geq -75 x_2 + x_4$$

$$1 \geq x_1$$

$$1 \geq x_2$$

SAMPLE SOLUTION

The data was saved in the file INTIN and is listed below:

*LIST INTIN

```
010 6 5
020 0 0 0 -1 -1 -1
030 180 20 30 1 2 2
040 150 30 20 2 1 2
050 0 -60 0 1 0 0
060 0 0 -75 0 1 0
070 1 1 0 0 0 0
080 1 0 1 0 0 0
```

READY

In the following run, method 3 is applied to the original problem, and prints the entire log every iteration starting from iteration 40.

```

INTLP
I=PRINT INSTRUCTIONS, O=DONT
=0
DATA FILE NAME
=INTIN
LSTART,L0FTEN,LMUCH,METH3D,CHANGES
=40 1 1 3 0

ITERATION 40 OBJ= -8.64583330E 01 DETERM.= 1.44E 03
X( 2)= 5.41666664E-01
X( 1)= 2.29166666E-01
X( 3)= 1.37500000E 01
X( 4)= 4.06250000E 01
X( 5)= 3.20833335E 01

ITERATION 41 OBJ= -8.58305540E 01 DETERM.= 3.00E 03
X( 2)= 6.63108736E-01
X( 1)= 4.93662443E-01
X( 3)= 2.96197464E 01
X( 4)= 4.97331553E 01
X( 5)= 6.47765177E 00

ITERATION 42 OBJ= -8.50000000E 01 DETERM.= 5.00E 01
X( 2)= 1.00000000E 00
X( 1)= 5.00000000E-01
X( 4)= 5.50000000E 01
X( 3)= 3.00000000E 01

ITERATION 43 OBJ= -8.50000000E 01 DETERM.= 1.10E 02
X( 2)= 8.18181820E-01
X( 1)= 3.63636363E-01
X( 4)= 6.13636365E 01
X( 5)= 1.81818181E 00
X( 3)= 2.18181818E 01

ITERATION 44 OBJ= -8.46491232E 01 DETERM.= 5.70E 01
X( 2)= 3.50877192E-01
X( 4)= 2.63157895E 01
X( 5)= 5.83333335E 01
X( 3)= 0.

```

Examining the results, it appears that x_2 might end up to be 1, and that the optimal objective value might be -85. The problem is restarted with x_2 constrained to be 1, and the objective function constrained to be greater than or equal to -85. Again, method 3 is used and the only heading of the log is to be printed every 20 iterations.

```

NEXT
=3
LSTART,LFTEN,LMUCH,METHOD,CHANGES
=20 20 2 3 2
VARIABLE #,VALUE
=0 -85
VARIABLE #,VALUE
=2 1

ITERATION 20 OBJ= -7.98113203E 01 DETERM.= 5.30E 01

```

OPTIMUM

```

ITERATION 28 OBJ= -7.60000000E 01 DETERM.= 1.00E 00
X( 4)= 54
X( 1)= 1
X( 3)= 22

```

We discover a feasible integer solution (1, 1, 22, 54, 0) with an objective value of -76. However, because of the constraint imposed on x_2 , it might not be optimal. We learn that any solution must have an objective value between -76 and -85. The problem is restarted using method 3 and including these two constraints on the objective value. Only the heading of the log is printed every 15 iterations.

```

NEXT
=3
LSTART,LFTEN,LMUCH,METHOD,CHANGES
=20 15 2 3 2
VARIABLE #,VALUE
=0 -85
VARIABLE #,VALUE
=-1 -76

ITERATION 20 OBJ= -8.49455557E 01 DETERM.= 3.95E 03
ITERATION 35 OBJ= -8.49164028E 01 DETERM.= 3.47E 03
ITERATION 50 OBJ= -8.27192984E 01 DETERM.= 5.70E 01
ITERATION 65 OBJ= -7.80000000E 01 DETERM.= 1.00E 02
ITERATION 80 OBJ= -7.70000000E 01 DETERM.= 9.00E 00

```

Good progress is made for 40 iterations, but then little change occurs. The iterations continue using method 5. Only the heading of the log is printed every 15 iterations.

INTLP-6

NEXT

=2

LOFTEN,LMUCH,METHOD

=15 2 5

ITERATION 95 OBJ= -7.70000000E 01 DETERM.= 1.70E 01

ITERATION 110 OBJ= -7.70000000E 01 DETERM.= 1.90E 01

ITERATION 125 OBJ= -7.70000000E 01 DETERM.= 5.80E 01

OPTIMUM

ITERATION 138 OBJ= -7.60000000E 01 DETERM.= 1.00E 00

X(2)= 1

X(4)= 54

X(3)= 22

X(1)= 1

NEXT

=1

*

Verification proves that the optimal solution is, in fact, the feasible solution found earlier.

This BASIC program aids in computing the cost improvement made possible through the more efficient layout of a plant, office, warehouse, store, etc.

METHOD

The Vollman-Ruml layout program is based on matrices established by inputting data defining:

- Plant configuration by department (logical divisions).
- Weighted figures for flow exchanges among departments (face-to-face contacts of personnel, order picking labor, material handling systems, including size and nature of items handled, distance traveled, etc.).
- Cost weightings for each department based on above measures.
- Starting layouts.

The program assumes the layout area can be described as rectangular. Also, "dummy", nonmovable departments can be utilized to make the area rectangular where required. More than one starting solution can be specified. A department can be designated as fixed (not movable). Such a fixed department is a washroom area with pipes and plumbing. Larger problems can be solved by increasing the dimension (DIM) statements from their current size of 40 departments.

SAMPLE PROBLEM

For detailed instructions, list the program:

*LIST 1-250

For sample data as described above, type:

*LIST 5000

SAMPLE RUN SOLUTION

```
SYSTEM 7BASIC
OLD OR NEW-OLD LAYOUT
READY
*RUN
```

LAYOUT WILL HANDLE THREE LAYOUT HEURISTICS:
CRAFT (=1), ANY IMPROVEMENT (=2), AND RANKED PRODUCT (=3).

LAYOUT-2

WHICH HEURISTIC 1, 2, OR 3. ?2

STARTING SOLUTION 1

INITIAL LAYOUT:

1	2	3
4	5	6

THE AVERAGE COST OF THE CURRENT LAYOUT IS 6383.75

FINAL LAYOUT

3	4	2
5	1	6

THE AVERAGE COST OF THE CURRENT LAYOUT IS 4881.25

STARTING SOLUTION 2

INITIAL LAYOUT:

3	2	1
4	5	6

THE AVERAGE COST OF THE CURRENT LAYOUT IS 6053.75

FINAL LAYOUT

3	2	6
4	1	5

THE AVERAGE COST OF THE CURRENT LAYOUT IS 5355

READY

*

This FORTRAN program computes the optimum solutions for linear programming problems. Specifically, a linear objective function is maximized (or minimized), subject to a set of linear constraints. Letting X_j refer to the structural variables, S_i to slack variables, and C_j and D_i to the objective function coefficients of the structural and slack variables, the objective function is expressed as:

$$Z = \sum_{j=1}^n C_j X_j + \sum_{i=1}^m D_i S_i$$

where

- n is the number of structural variables
- m is the number of constraints

Each constraint is of the form:

$$\sum_{j=1}^n A_{ij} X_j \leq (\text{or } = \text{ or } \geq) B_i$$

where

A_{ij} refers to the structural variable coefficients, and B_i refers to the requirements columns or right-hand side values.

The results of the LP solution consist of an optional iteration log, the basic variable results, and the nonbasic variable results. The structural variables are identified by the number 1 through N . The slack variables associated with each constraint are identified by the numbers 101 through 100+ M . Variable identifications 999 and 200+($M+1$) are added by the program to handle the "greater than" constraints and do not normally appear in the basic and nonbasic variable results.

The iteration log, which is selected at run time, consists of the iteration count, the identifications of the variables entering and leaving the basis, and the current value of the objective function.

The basic variable results consist of the variable identification, the objective function coefficient, and the answer (value of the variable) for each variable in the final solution.

The nonbasic variable results consist of the variable identification, the objective function coefficient, and the answer (reduced cost or shadow price) for each variable not in the final solution.

An unfeasible condition is indicated by a slack variable associated with an "equal to" constraint and/or the slack variable identified as $200+(M+1)$ appearing in the basic variable results at other than a zero value. This means that the problem contains two or more constraints that cannot be simultaneously satisfied.

The appearance of a negative identification for a structural variable in the nonbasic results indicates that this variable is unbounded. None of the problem constraints restrict this variable from entering the solution at an infinite value.

INSTRUCTIONS

At execution, the program asks for the data file name. It is assumed this file has been built without line numbers. The first line of data is used for problem identification. The second line contains m (the number of constraints), n (the number of structural variables) and either MAX or MIN indicating whether the problem is a maximization or minimization problem. The next line is the objective function coefficients, C_j . The data for each constraint follows in the following order:

$$A_{1j} \ A_{12} \ . \ . \ . \ A_{in} < B_i \ D_i$$

where "<" can be replaced by ">" or "=" to indicate the sense of the constraint.

The data for additional problems may also be included sequentially in the same file.

RESTRICTIONS

$$\begin{aligned} m &\leq 30 \\ n &\leq 50 \end{aligned}$$

where

- m is the number of constraints
- n is the number of structural variables

SAMPLE PROBLEM

Maximize the function:

$$Z = -X - Y - Z + W$$

subject to the constraints:

$$-X - 1Y + 4Z - 1W \geq 5$$

$$2Y - Z - W \geq 4$$

$$X - 3Y + 4Z - 4W \leq 5$$

$$X - 2Y \leq 3$$

SAMPLE SOLUTION

The data for this problem was stored in the file LPDATA.

*LIST LPDATA

TEST PROBLEM FOR LP

```
4 4 MAX
-1 -1 -1 1
-1 -1 4 -1 > 5 0
0 2 -1 -1 > 4 0
1 -3 4 -4 < 5 0
1 -2 0 0 < 3 0
```

READY

*RUN

DATA FILE NAME

= LPDATA

TEST PROBLEM FOR LP

4 ROWS X 4C0LS

1=PRINT ITERATION LOG, 2=SUPPRESS

= 2

OBJ. FUNCT. = -5.00000E+00

BAS VAR	OBJ. COEFF.	RESULT
2	-1.00000E+00	3.00000E+00
3	-1.00000E+00	2.00000E+00
103<	0.	6.00000E+00
104<	0.	9.00000E+00

N. BAS VAR	OBJ. COEFF.	RESULT
1	-1.00000E+00	1.42857E+00
4	1.00000E+00	1.42857E-01
101>	0.	4.28571E-01
102>	0.	7.14286E-01

*

This BASIC program solves small linear programming problems by a simple version of the SIMPLEX method. For similar programs, see LINPRO and LNPROG.

INSTRUCTIONS

For detailed instructions, list the program:

*LIST 1000-1710

Before using the program, arrange all constraints (linear restrictions on the problem variables) as follows:

1. The "less than or equal" inequalities
2. The strict equalities
3. The "greater than or equal" inequalities

To use the program, starting on line 10000, enter the data in the following order:

1. Coefficients for each of the problem variables (activities). Include zeros for variables not appearing in each restriction, starting with the first restriction, and proceeding in order until all coefficients of all restrictions have been entered in data statements.
2. Elements of the "B" vector (constants comprising the right side of all restrictions) in the same order as the restrictions.
3. The coefficients of the (linear) objective function, in the same order as the restrictions, including zeroes if needed.

Then type RUN.

REFERENCES

Reinfeld and Vogel, Mathematical Programming, Englewood Cliffs, New Jersey, Prentice-Hall, Inc., 1958.

SAMPLE PROBLEM

Maximize the function: $Z = 4X_1 + 4X_2 + 3X_3$, while satisfying the following constraints:

$$-1X_1 + 2X_2 + 3X_3 \leq 15$$

$$0X_1 - 1X_2 + 1X_3 \leq 4$$

$$2X_1 + 1X_2 - 1X_3 \leq 6$$

$$1X_1 - 1X_2 + 2X_3 \leq 10$$

$$1X_1 + 0X_2 + 0X_3 \leq 8$$

$$0X_1 + 1X_2 + 0X_3 \leq 4$$

$$0X_1 + 0X_2 + 1X_3 \leq 4$$

$$0X_1 + 0X_2 + 1X_3 \geq 3$$

SIMPLEX-2

Beginning on line 10000, enter the above constraints in the following order:

```
10000 DATA -1,2,3,0,-1,1,2,1,-1,1,-1,2
10001 DATA 1,0,0,0,1,0,0,0,1,0,0,1
10002 DATA 15,4,6,10,8,4,4,3
10003 DATA 4,4,3
```

As the program runs, MAX is typed for the function, and the value 3 is typed for the number of activities (variables). When the number of constraints is called for, 7,0,1 is typed: 7 is the number of $\leq$ restrictions, zero is the number of $=$ restrictions, and one is the number of $\geq$ restrictions.

NOTE: The problem solution is:

Variable (activity) $X_1 = 3.4$

Variable (activity) $X_2 = 3.2$

Variable (activity) $X_3 = 4.0$

SAMPLE SOLUTION

```
P
SYSTEM ?BASIC
OLD OR NEW-OLD SIMPLEX
READY
*RUN
```

LIST THIS PROGRAM-1000-1710 FOR INSTRUCTIONS****

```
READY
*10000 DATA -1,2,3,0,-1,1,2,1,-1,1,-1,2
*10001 DATA 1,0,0,0,1,0,0,0,1,0,0,1
*10002 DATA 15,4,6,10,8,4,4,3
*10003 DATA 4,4,3
*RUN
```

```
IS THIS A MAX OR A MIN PROBLEM ?MAX
HOW MANY ACTIVITIES DOES YOUR PROBLEM HAVE ?3
HOW MANY CONSTRAINTS IN YOUR PROBLEM ( $\leq$ ,  $=$ ,  $\geq$ ) ?7,0,1
DO YOU WANT THE INITIAL TABLEAU PRINTED ?N
DO YOU WANT THE INTERMEDIATE BASIC SOLUTIONS PRINTED ?YES
DO YOU WANT THE FINAL TABLEAU PRINTED ?YES
```

STRUCTURAL ACTIVITIES	1 -	3
SURPLUS ACTIVITIES	4 -	4
SLACK ACTIVITIES	5 -	11
ARTIFICIAL ACTIVITIES	12 -	12

PHASE I INITIATED
=====

INITIAL BASIC SOLUTION

ACTIVITY	VALUE
5	15
6	4
7	6
8	10
9	8
10	4
11	4
12	3

(Z= -3)

BASIC SOLUTION AFTER ITERATION 1

ACTIVITY	VALUE
3	3
5	6
6	1
7	9
8	4
9	8
10	4
11	1

(Z= 0)

PHASE II INITIATED AFTER 1 ITERATION
=====

INITIAL BASIC SOLUTION

ACTIVITY	VALUE
3	3
5	6
6	1
7	9
8	4
9	8
10	4
11	1

(Z= 9)

BASIC FEASIBLE SOLUTION AFTER ITERATION 2

ACTIVITY	VALUE
1	4
3	3
5	10
6	1
7	1
9	4
10	4
11	1

(Z= 25)

SIMPLEX-4

BASIC FEASIBLE SOLUTION AFTER ITERATION 3

ACTIVITY	VALUE
1	4.333333
2	.3333333
3	3
5	9.666667
6	1.333333
9	3.666667
10	3.666667
11	1

(Z= 27.66667)

BASIC FEASIBLE SOLUTION AFTER ITERATION 4

ACTIVITY	VALUE
1	4
2	2
3	4
4	1
5	3
6	2
9	4
10	2

(Z= 36)

BASIC FEASIBLE SOLUTION AFTER ITERATION 5

ACTIVITY	VALUE
1	3.4
2	3.2
3	4
4	1
6	3.2
8	1.8
9	4.6
10	.8

(Z= 38.4)

THIS SOLUTION IS MAXIMAL.

OPTIMAL DUAL SOLUTION

CONSTRAINT	DUAL EVALUATOR
1	.8
2	0
3	2.4
4	0
5	0
6	0
7	3
8	0

OPTIMAL TABLEAU

X(1)	X(2)	X(3)	X(4)	X(5)
0	0	0	0	.6
0	0	0	0	.4
0	1	0	0	.4
1	0	0	0	-.2
0	0	0	0	.2
0	0	0	0	-.4
0	0	0	1	0
0	0	1	0	0
OBJ FN ROW:				
0	0	0	0	.8

X(6)	X(7)	X(8)	X(9)	X(10)
0	-.2	1	0	0
1	.2	0	0	0
0	.2	0	0	0
0	.4	0	0	0
0	-.4	0	1	0
0	-.2	0	0	1
0	0	0	0	0
0	0	0	0	0
OBJ FN ROW:				
0	2.4	0	0	0

X(11)	X(12)	B-VECTOR
-4	0	1.8
-2	0	3.2
-1	0	3.2
1	0	3.4
-1	0	4.6
1	0	.8
1	-1	1
1	0	4
OBJ FN ROW:		
3	0	38.4

READY

*

NOTE: Only the final or optimal tableau was requested in this run. The beginning or initial can be printed if the user desires. See requests at beginning of run.

This FORTRAN program provides solutions to a wide range of time-series forecasting problems. The program follows four fundamental steps to provide useful predictions:

1. Cyclic analysis of past data
2. Trend analysis of past data
3. An error analysis for comparing forecast with actual data
4. A synthesis of analyses to form a forecast

INSTRUCTIONS

Data Preparation

The data can be entered either from a data file or from the keyboard. In either case the format is identical. The data file format is described. The file may be given any name and it can be built with or without line numbers. The first line of the file is alphanumeric title information. The second line contains the following parameters, separated by commas:

<u>VARIABLE NAME</u>	<u>DESCRIPTION</u>
L	LEAD TIME The number of time periods for which the forecast is to be calculated and the forecasting parameters optimized. Lead time is usually specified as the minimum length of time desired to accurately forecast the future.
IH	FORECAST HORIZON The number of time periods for which the forecast is to be projected, regardless of lead time and accuracy.
I6	I6 = 0 - all three types of smoothing to be used by program. I6 = 1 - single smoothing to be used by program. I6 = 2 - double smoothing to be used by program. I6 = 3 - triple smoothing to be used by program.
Y-SMALL	Smallest ordinate for plot of forecast.
Y-LARGE	Largest ordinate for plot of forecast.

The historical data is entered next. This data consists of a raw data point and optional base series point for each time period. A base is a time series of values which are used to adjust (transform) both the raw data and the forecast. Typically, a base series may represent human judgment, cyclic variations, results of a multiple regression correlation analysis, or known phenomena. The base series is usually used to transform the raw data into a new time series, which, in turn, is used for intermediate computations. The results are retransformed by the base series to form a forecast.

The raw data and base series data points (if there is a base series) are entered on the following lines, with one raw data point and one base series data point per line. Data points are entered in free-field format.

The termination of both the raw data and base series data points is specified with a final data point greater than or equal to 1E15. If there is no base series, then the first data line must have a value for the first base series data point greater than or equal to 1E15.

The program can handle up to 310 raw data points and 450 base series data points.

The final line in the data file contains up to eight trial smoothing constants, ALPHA, entered in free-field format. Additional ALPHA's may be specified during execution of the program. These constants must be between 0 and 1. The larger the constant, the more weight is given to the recent history in calculating the forecast.

Execute Instructions

To execute the program, type:

LIB TCAST
RUN

The program will ask for the names of an input and output file and an optional print-out level indicator.

The input filename is the name of a previously prepared data file. If the data is to be entered from the terminal, enter blanks for the file name. There are seven groups of output data. For each group, there is the option of writing the output on the specified output file, or printing it at the terminal. The print option parameters are:

- COMPLETE — Print all seven (7) groups.
- ASK — Give the output option on all seven groups of output.
- PART — Give the output option for only CYCLIC ERRORS, TREND ANALYSIS, and FORECAST DATA. All other output is to be written on the output file.
- LEAST — Print CYCLIC ERRORS and TREND ANALYSIS at the terminal; all other output is to be written on the output file.

If none of the above is entered, PART is assumed. The output file can be listed at the terminal, printed at the central site (using BPRINT), or examined using EDIT.

Output Description

A description of the output follows:

- INITIAL DATA
The raw data points, base series, etc., as read from the data file. This output is useful to ascertain that the data was correctly entered.
- CYCLIC ERRORS
The cyclic error $ERR(K)$ is a relative measure of residual variance for a cycle of length K .
The local minima of cyclic error indicate significant cyclic behavior corresponding to that cycle length. This type of analysis is useful to determine which cyclic intervals it would be meaningful to force, and other harmonics.
The program prints the cycle length which minimizes the relative error. The user then selects the period to be used.
- CYCLIC VALUES
The cyclic values give a quantitative description of the shape of the cycle.
- CYCLIC RESIDUES
The residues remaining after the raw data is corrected for both base series, and cycle series are output.
At this point the period can be changed and the cyclic values and cyclic residues recalculated.
- TREND ANALYSIS
The mean absolute deviations (MAD) which are associated with each smoothing constant and type of smoothing are printed.
After the program has performed this analysis for each smoothing constant in the data file, additional constants can be entered from the terminal. A null response (carriage return) signifies there are no more constants to be entered. Specify the smoothing type and smoothing constant to be used for the forecast.

- FORECAST

Specify the time period for which the forecast output is to begin (the output cannot begin before time period = lead time + 5). For each time period the output consists of:

- Forecast of the residue, which is the forecast of raw data point minus base series data point minus cyclic value.
- Composite forecast, which equals forecast of the residue plus base series data point plus cyclic value.
- Raw data point.
- Error in the forecast, which equals raw data point minus composite forecast.
- A plot of the composite forecast (.) and raw data point (*) versus time for each time period. When the plot of the composite forecast and the raw data point occur in the same print position, an "=" is printed.

After the end of the historical data, there are no errors. Beyond this point, only the time period, forecast of the residue, and composite forecast are printed. These time periods are of prime interest, since they constitute the forecast.

The forecast plot can be directed to the terminal, the main output file, or a distinct plot file. If it is directed to the main output file, then each data line will be followed by the plot line.

- STATISTICAL INFORMATION

- S1, S2, S3, the exponentially smoothed variable for single, double, and triple smoothing, respectively, followed by CED1, CED2, CED3, the current expected demand for single, double, and triple smoothing, respectively.
- C2, C3, the change per unit time in exponentially smoothed average for double and triple smoothing, respectively, and finally RC3 the rate of change per unit time in the exponentially smoothed average for triple smoothing.
- The time period after which the forecast is not within the mean absolute deviation (MAD) limits.
- Linear least square curve fit.
- Mean.
- Variance.

NOTE: The user can give a null response to any question, in which case the program chooses the best option or parameter.

REFERENCES

Series G-200/400/600/6000 Time Series Forecasting Implementation Guide, Order No. BQ07.

SAMPLE PROBLEM

Perform a time series forecast on the following data with a lead time of 3 and a horizon of 6.

data point	1.01	2	3.02	4	5.01	6	1	2	3.02	2	3	4					
base point				3	3		3	0	0	0		1	1	1	3	3	3

SAMPLE SOLUTION

The data was entered from the terminal. However, the data could have been entered in a data file as below:

SAMPLE 1 - DATA FILE

```

3, 6, 0, 0., 10.
1
2
3
4, 3
5, 3
6, 3
1, 0
2, 0
3, 0
2, 1
3, 1
4, 1
1E15, 3
3
3
1E15
.1 .2 .3

```

During execution, part of the data was directed to the output file DUMP. This file is also listed.

SYSTEM ?YFOR
 OLD OR NEW-LIB TCAST
 READY
 *RUN

ENTER '?' FOR INSTRUCTIONS
 FILES (PRINT OPTION)
 =DUMP

ENTER PROBLEM TITLE-
 =SAMPLE PROBLEM - ENTERING DATA FROM TERMINAL

ENTER LEAD TIME, HORIZON, SMOOTHING TYPE, YSMALL, YLARGE-
 =3,6,0,0,10

ENTER DATA, BASE POINTS(ONE PAIR/LINE)

=1.01
 =2
 =3.02
 =4,3
 =5.01,3
 =6,3
 =1
 =2
 =3.02
 =2,1
 =3,1
 =4,1
 =1E15,3
 =3
 =3
 =1E15

DIRECT CYCLIC ERROR TO FILE(Y OR N)-
 =N

CYCLIC ERROR

K	ERR(K)
1	0.166615
2	0.196793
3	0.
4	0.301417
5	0.337813
6	0.

PERIOD OF MOST DOMINANT CYCLE= 3

PERFORM ANALYSIS FOR PERIOD-
=3

DIRECT TREND ANALYSIS TO FILE-
=N

TREND ANALYSIS

ENTER ALPHAS (MAX OF 8)
=.1, .2, .3

ALPHA	TYP	SM	ERROR MAD
0.10000	1		0.00995
0.10000	2		0.00882
0.10000	3		0.00801
0.20000	1		0.00929
0.20000	2		0.00752
0.20000	3		0.00689
0.30000	1		0.00862
0.30000	2		0.00689
0.30000	3		0.00968

ADDITIONAL ALPHAS-
=.25, .4

0.25000	1		0.00894
0.25000	2		0.00693
0.25000	3		0.00829
0.40000	1		0.00820
0.40000	2		0.00895
0.40000	3		0.01225

ADDITIONAL ALPHAS-
=

OPTIMUM SMOOTHING TYPE=3 ALPHA=0.20000000

WHAT SMOOTHING TYPE AND ALPHA-
=3, .2

DIRECT FORECAST PLOT TO TERMINAL, OUTPUT FILE, OR PLOT FILE (T,O,P)
=T

FORECAST PLOT

BEGIN FORECAST AT PERIOD-
=0

TCAST-8

TIME 0.

0.10000E 02

8	=				
9		=			
10	=				
11		=			
12			=		
13				.	
14					.
15					.
16	.				
17		.			
18			.		

CAUTION, FORECAST NOT WITHIN MAD LIMITS AFTER TIME 15

*LIST DUMP

PROBLEM NAME:
SAMPLE PROBLEM - ENTERING DATA FROM TERMINAL

INITIAL DATA

NUMBER OF RAW DATA POINTS-- 12
NUMBER OF BASE DATA POINTS-- 15
FORECAST HORIZON-- 6
LEAD TIME-- 3

TIME	RAW DATA
1	1.01000
2	2.00000
3	3.02000
4	4.00000
5	5.01000
6	6.00000
7	1.00000
8	2.00000
9	3.02000
10	2.00000
11	3.00000
12	4.00000

TIME	BASE SERIES	RESIDUE
1	0.	1.01000
2	0.	2.00000
3	0.	3.02000
4	3.00000	1.00000
5	3.00000	2.01000
6	3.00000	3.00000
7	0.	1.00000
8	0.	2.00000
9	0.	3.02000
10	1.00000	1.00000
11	1.00000	2.00000
12	1.00000	3.00000
13	3.00000	-3.00000
14	3.00000	-3.00000
15	3.00000	-3.00000

CYCLIC VALUES

PERIOD= 3

T	C(T)
1	0.999280
2	1.007680
3	-2.015200

CYCLIC RESIDUES

TIME	RESIDUE
1	1.01000
2	1.00072
3	1.01304
4	1.00824
5	1.01896
6	1.00128
7	1.01648
8	1.01720
9	1.02952
10	1.02472
11	1.02544
12	1.01776

FORECAST DATA

USED ALPHA	TYP SM			
0.20000	3			
TIME	RESIDUE	COMPOSITE	ACTUAL	ERROR
		(.)	(*)	
8	1.0180	2.0008	2.0000-0.80594E-03	
9	1.0082	2.9987	3.0200 0.21312E-01	
10	1.0142	1.9895	2.0000 0.10523E-01	
11	1.0182	2.9928	3.0000 0.72432E-02	
12	1.0299	4.0122	4.0000-0.12168E-01	
13	1.0325	3.9995		
14	1.0338	5.0001		
15	1.0277	6.0017		
16	1.0292	0.98798		
17	1.0307	1.9888		
18	1.0323	2.9980		

STATISTICAL INFORMATION

S1= 1.01866
 S2= 1.01455
 S3= 1.01141

CED1= 1.01866
 CED2= 1.02278
 CED3= 1.02375

C2= 0.00103
 C3= 0.00124
 RC3= 0.00006

LEAST SQUARES CURVE FIT

Y= 2.645+ 0.055*X

MEAN= 3.005
 VARIANCE= 2.167

READY

*

This FORTRAN subroutine plots a maximum of nine curves simultaneously.

INSTRUCTIONS

The calling sequences are:

```
CALL PLOT1 (NCURVES, YMIN, YMAX, NPOS, XSTART, XDELTA, MARKS)
```

```
CALL PLOT (NCURVES, Y)
```

where

- PLOT1 is called first to initialize the plot parameters, and PLOT is called each time a line of PLOT is to be printed.
- NCURVES is the number of curves to be simultaneously plotted.
- YMIN, YMAX are the minimum and maximum Y values.
- NPOS is the number of character positions available for the plot.
- XSTART is the starting X value of the plot. The first to PLOT transmits the Y values corresponding to XSTART.
- XDELTA is the amount by which X is to be incremented on each call to PLOT.
- MARKS is the character*1 array of the marks (such as *, +, .) to be used in marking the Y values on each function plotted.
- Y is the array of Y values to be plotted at any particular call to PLOT.

SAMPLE PROBLEM

Plot the following curves:

$$Y1 = 3e^{-X/4}$$

$$Y2 = \sin(\pi X/2)$$

$$Y3 = 3e^{-X/4} \sin(\pi X/2)$$

$$Y4 = -3e^{-X/4}$$

where

$$YMAX = 4.0, \quad YMIN = -4.0,$$

and the number of curves, $N = 4$

PLOT1-2

*LIST

```

010 CHARACTER*1 MARKS(4)/"X", "#", "+", "$" /
020 DIMENSION Y(4)
030 N=4
040 YMAX=4.
050 YMIN=-4.
060 DX=.25
070 CALL PLOT1(N,YMIN,YMAX,72,-1.75,DX,MARKS)
080 DO 100 J=1,50
090 X=J*DX-2.
100 Y(1) = 3.* EXP(-.25*
      X)
110 Y(2) = SIN(1.5705*X)
120 Y(3) = Y(1) * Y(2)
130 Y(4) = -Y(1)
140 CALL PLOT(N,Y)
150 100 CONTINUE
160 STOP
170 END

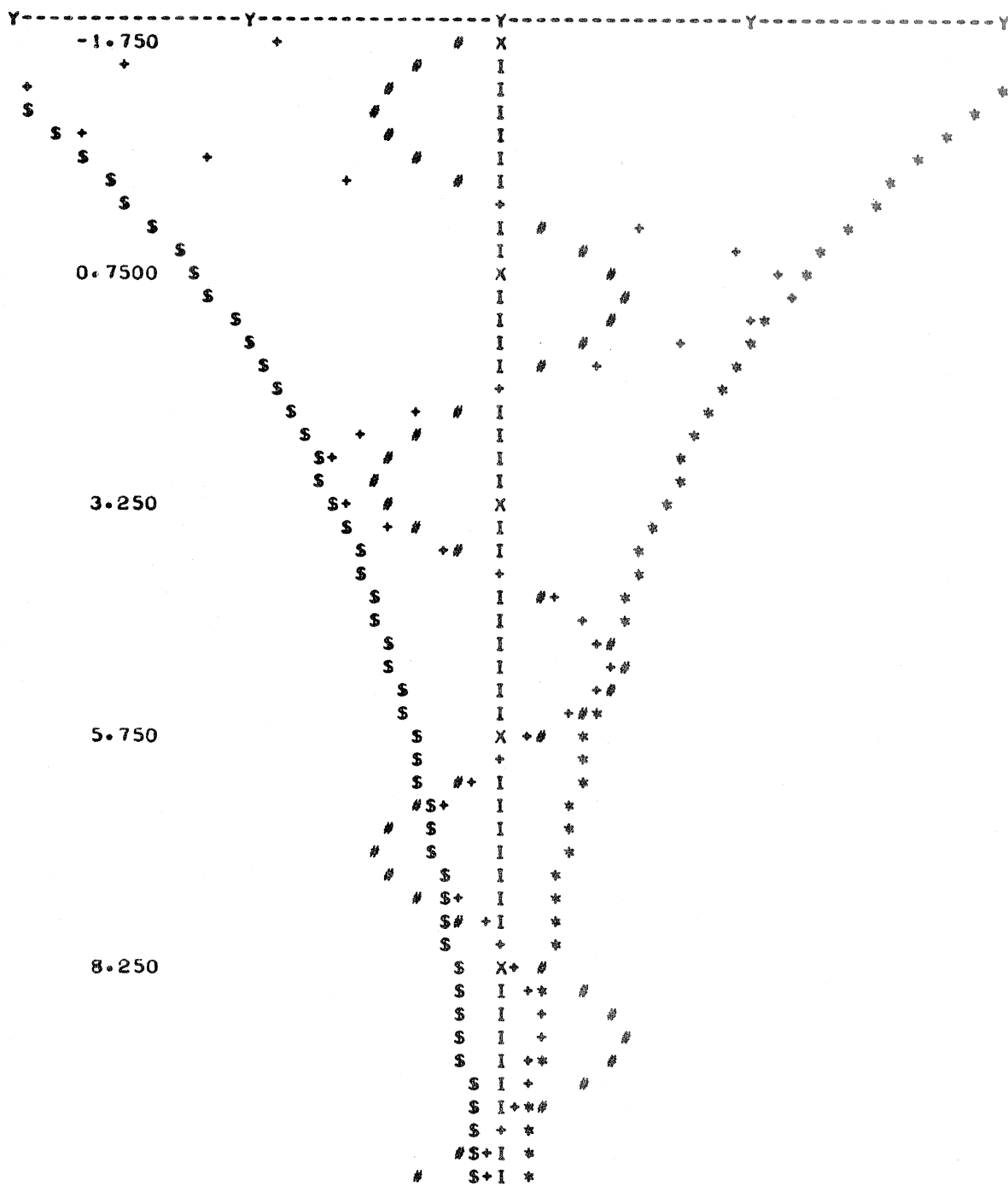
```

READY

*RUN *;PLOT1=(CORE=19)

Y-AXIS MARKS ARE--

Y 1	=	-4.000
Y 2	=	-2.000
Y 3	=	0.
Y 4	=	2.000
Y 5	=	4.000



*

This FORTRAN program is the driver program used, in conjunction with the pre-processor PREPRS, to run a translated lesson in the EXPER language (see Series G-200 Time-Sharing EXPER Language, Order No. BS05). It administers the lesson and writes the students' responses on a specified file. EXPER is a Computer Assisted Instruction (CAI) language.

INSTRUCTIONS

To use the DRIVES subroutine, write a four-line program as follows:

```
10*#RUN *;LIBRARY/DRIVES,R = (CORE=19)
20 CALL DRIVER ("userid/lesson;", "userid/respon;")
30 STOP
40 END
```

where

- userid/lesson is the catalog/file description of the file containing the translated lesson.
- userid/respon is the catalog/file description of the student response file.

NOTE: Both file descriptions require a semicolon (;) to flag the end of the description.

SAMPLE PROBLEM

A sample lesson was previously translated and stored in the file LESSON (see PREPRS Sample Problem).

SAMPLE RUN

A student response file was created. The four-line driver program was written and executed. Then the response file was listed.

```
*10*#RUN *;LIBRARY/DRIVES,R=(CORE=19)
*20 CALL DRIVER("EXOTIC/L1;", "EXOTIC/RESPON;")
*RUN
HELLO AND WELCOME TO SERIES 600/6000 TIME SHARING. THIS SEQUENCE
OF PROGRAMS IS DESIGNED TO HELP YOU LEARN TO WRITE INSTRUCTION-
AL MATERIALS IN THE EXPR COMPUTER LANGUAGE.
```

FIRST, LETS ESTABLISH THE METHOD BY WHICH WE WILL COMMUNICATE.
WHEN I WISH TO HAVE YOU RESPOND TO A QUESTION OR STATEMENT,
I WILL TYPE AN EQUAL SIGN (=) AND THEN STOP, WAITING FOR
YOUR RESPONSE.

LETS TRY IT. WHEN I TYPE AN EQUAL SIGN AND STOP, YOU TYPE
YOUR NAME AND THEN PRESS THE RETURN KEY TO LET ME KNOW THAT
YOU ARE FINISHED. (THE RETURN KEY IS LOCATED AT THE FAR RIGHT
HAND SIDE OF THE KEYBOARD.)

=HONEYWELL

GOOD. THAT'S HOW WE WILL COMMUNICATE.

```
*NEW
READY
*100-A
*110 WHICH STATEMENT BEST DESCRIBES THE CAPABILITIES
*120 OF EXPR?
*130 A. A LANGUAGE DESIGNED FOR SCIENTIFIC CALCULATIONS.
*140 B. A LANGUAGE DESIGNED FOR BUSINESS APPLICATIONS.
*150 C. A LANGUAGE DESIGNED FOR USE BY INSTRUCTORS
*160 WITH A NEED TO WRITE LESSONS TO BE ADMINISTERED
*170 BY A COMPUTER.
*180 D. ALL OF THE ABOVE.
*190:INP
*200:M21:C:
*210:JK2:ON:
*220 WRONG. EXPR IS A C.A.I.(COMPUTER ASSISTED INSTRUCTION)
*230 AUTHOR LANGUAGE. WITH THIS INFORMATION TRY THE QUESTION
*240 AGAIN.
*250
*260:JK2:A1000:A:
*270-ON
*280 THATS RIGHT. EXPR IS DESIGNED FOR EASY LESSON WRITING
*290 WITH CAPABILITIES FOR MATCHING RESPONSES AND BRANCHING
*300 ON THE VALUE OF SCORE COUNTERS.
*310:*ND
*SAVE SOURCE
DATA SAVED--SOURCE
*NEW
READY
*10*#RUN *;LIBRARY/PREPRS,R=(CORE=19)
*20 CALL PREPR0("EXOTIC/SOURCE;", "EXOTIC/LESSON;")
*30 STOP
*40 END
*SAVE LESSON
DATA SAVED--LESSON
*RUN
100 100-A
270 117-ON
END PASS 1
```

```

*NEW
READY
*SAVE RESPF
DATA SAVED--RESPF
*10*#RUN *;LIBRARY/DRIVES,R=(CORE=19)
*20 CALL DRIVER("EXOTIC/LESSON;", "EXOTIC/RESPF;")
*30 STOP
*40 END
*RUN
WHICH STATEMENT BEST DESCRIBES THE CAPABILITIES
OF EXPER?
  A. A LANGUAGE DESIGNED FOR SCIENTIFIC CALCULATIONS.
  B. A LANGUAGE DESIGNED FOR BUSINESS APPLICATIONS.
  C. A LANGUAGE DESIGNED FOR USE BY INSTRUCTORS
    WITH A NEED TO WRITE LESSONS TO BE ADMINISTERED
    BY A COMPUTER.
  D. ALL OF THE ABOVE.

```

=B
 WRONG. EXPER IS A C.A.I. (COMPUTER ASSISTED INSTRUCTION)
 AUTHOR LANGUAGE. WITH THIS INFORMATION TRY THE QUESTION
 AGAIN.

```

WHICH STATEMENT BEST DESCRIBES THE CAPABILITIES
OF EXPER?
  A. A LANGUAGE DESIGNED FOR SCIENTIFIC CALCULATIONS.
  B. A LANGUAGE DESIGNED FOR BUSINESS APPLICATIONS.
  C. A LANGUAGE DESIGNED FOR USE BY INSTRUCTORS
    WITH A NEED TO WRITE LESSONS TO BE ADMINISTERED
    BY A COMPUTER.
  D. ALL OF THE ABOVE.

```

=C
 THATS RIGHT. EXPER IS DESIGNED FOR EASY LESSON WRITING
 WITH CAPABILITIES FOR MATCHING RESPONSES AND BRANCHING
 ON THE VALUE OF SCORE COUNTERS.

*

EXPERn-2

HELLO AND WELCOME TO SERIES 600/6000 TIME SHARING. THIS SEQUENCE OF PROGRAMS IS DESIGNED TO HELP YOU LEARN TO WRITE INSTRUCTIONAL MATERIALS IN THE EXPER COMPUTER LANGUAGE.

FIRST, LETS ESTABLISH THE METHOD BY WHICH WE WILL COMMUNICATE. WHEN I WISH TO HAVE YOU RESPOND TO A QUESTION OR STATEMENT, I WILL TYPE AN EQUAL SIGN (=) AND THEN STOP, WAITING FOR YOUR RESPONSE.

LETS TRY IT. WHEN I TYPE AN EQUAL SIGN AND STOP, YOU TYPE YOUR NAME AND THEN PRESS THE RETURN KEY TO LET ME KNOW THAT YOU ARE FINISHED. (THE RETURN KEY IS LOCATED AT THE FAR RIGHT HAND SIDE OF THE KEYBOARD.)
= HONEYWELL

GOOD. THAT'S HOW WE WILL COMMUNICATE.

This FORTRAN program is the preprocessor used, in conjunction with DRIVES, to process and run the EXPER language (see Series G-200 Time-Sharing EXPER Language, Order No. BS05). This subroutine translates an EXPER source program and saves the translated lesson on a specified file. EXPER is a Computer Assisted Instruction (CAI) language.

INSTRUCTIONS

To use this subroutine, write a four-line program as follows:

```
10*#RUN *; LIBRARY/PREPRS,R = (CORE = 19)  
20 CALL PREPRO ('userid/source;', 'userid/lesson;')  
30 STOP  
40 END
```

where

- userid/source is the catalog/file description of the file in which the EXPER source program is saved.
- userid/lesson is the catalog/file description of the file which will receive the translated lesson.

NOTE: Both file descriptions require a semicolon (;) to flag the end of the description.

SAMPLE PROBLEM

A sample lesson was written and saved in the file SOURCE. Translate this lesson prior to execution of the driver (see DRIVES Sample Problem).

SAMPLE RUN

The translated lesson is saved in the file LESSON.

*NEW

READY

*100-A

*110 WHICH STATEMENT BEST DESCRIBES THE CAPABILITIES

*120 OF EXPER?

*130 A. A LANGUAGE DESIGNED FOR SCIENTIFIC CALCULATIONS.

*140 B. A LANGUAGE DESIGNED FOR BUSINESS APPLICATIONS.

*150 C. A LANGUAGE DESIGNED FOR USE BY INSTRUCTORS

*160 WITH A NEED TO WRITE LESSONS TO BE ADMINISTERED

*170 BY A COMPUTER.

*180 D. ALL OF THE ABOVE.

*190:INP

*200:M21:C:

*210:JK2::ON:

*220 WRONG. EXPER IS A C.A.I.(COMPUTER ASSISTED INSTRUCTION)

*230 AUTHOR LANGUAGE. WITH THIS INFORMATION TRY THE QUESTION

*240 AGAIN.

*250

*260:JK2;A;@@@:A:

*270-ON

*280 THATS RIGHT. EXPER IS DESIGNED FOR EASY LESSON WRITING

*290 WITH CAPABILITIES FOR MATCHING RESPONSES AND BRANCHING

*300 ON THE VALUE OF SCORE COUNTERS.

*310:*ND

*SAVE SOURCE

DATA SAVED -- SOURCE

*NEW

READY

10#RUN *;LIBRARY/PREPRS,R=(CORE=19)

*20 CALL PREPRO ("EXOTIC/SOURCE;", "EXOTIC/LESSON;")

*30 STOP

*40 END

*SAVE LESSON

DATA SAVED -- LESSON

*RUN

100 100-A

270 117-ON

END PASS 1

This BASIC program is a simulated card game of Las Vegas-type blackjack.

INSTRUCTIONS

For instructions run the program.

SAMPLE SOLUTION

This is an actual demonstration game conducted briefly to show some of the points of the game.

* RUN

BLKJAK

THIS DEMONSTRATION SHOWS THE VERSATILITY OF TIME-SHARING BY SIMULATING A GAME OF BLACKJACK. DO YOU NEED INSTRUCTIONS (1=YES,0=NO) ? 1

HERE ARE THE LAS VEGAS RULES FOR PLAYING BLACKJACK:

> WAGER: THE HOUSE LIMIT IS \$500, SO TYPE IN A NUMBER FROM 0 TO 500. TO TERMINATE GAME, ENTER ZERO. -

> THE DEAL: I DEAL MYSELF 2 CARDS AND SHOW YOU ONE. THEN I DEAL YOU TWO CARDS, AND ASK IF YOU WANT A HIT (ANOTHER CARD). YOU HAVE SEVERAL OPTIONS DEPENDING ON THE CARDS YOU HOLD AND MY UP CARD:

- * STAND - BY TYPING A ZERO
- * TAKE A HIT - BY TYPING A ONE
- * GO DOWN FOR DOUBLES - BY TYPING A TWO
- * SPLIT A PAIR - BY TYPING A THREE

> INSURANCE: IF MY UP CARD IS AN ACE, I WILL ASK IF YOU WANT INSURANCE. IF YOU DO TYPE A ONE, BETTING ONE-HALF OF YOUR WAGER THAT I DO HAVE BLACKJACK. IF I DO, I PAY 2-TO-1 ON YOUR INSURANCE BET. YOU LOSE YOUR ORIGINAL WAGER SINCE I HAVE BLACKJACK, SO WE ARE EVEN FOR THE HAND. IF I DON'T HAVE BLACKJACK, YOU LOSE YOUR INSURANCE BET AND THE GAME CONTINUES.

IF YOU REFUSE INSURANCE (BY TYPING A ZERO) THE GAME CONTINUES AS NORMAL.

> THE PLAY: WHEN YOU FINALLY STAND (BY TYPING A ZERO) I WILL DRAW CARDS UNTIL:

- * I HAVE AT LEAST A HARD 17 (HARD MEANS THE TOTAL DOES NOT INCLUDE AN ACE BEING COUNTED AS 11)
- * I HAVE A SOFT 18 (SOFT MEANS THE TOTAL INCLUDES AN ACE COUNTED AS 11)
- * I REACH A TOTAL OF 21
- * I EXCEED 21 AND BUST

> ITEMS:

- * I PAY 1.5-TO-1 ON BLACKJACK
- * I DON'T RECOGNIZE 5-CARDS-AND-UNDER
- * YOU MAY DOUBLE DOWN ON A SPLIT HAND
- * YOU DON'T LOSE ON A TIE HAND...WE PUSH

<<<GOOD LUCK>>>

BLKJAK-2

THE 600 IS THE DEALER AND GETS A BREAK AT 1945 HOURS. WHAT
TIME IS IT NOW ?300

WAGER ?50

I SHOW ACE OF DIAMONDS
FIRST CARD IS JACK OF HEARTS
NEXT CARD IS 2 OF DIAMONDS
INSURANCE ANYONE (TYPE 1 OR 0) ?1
YOU WIN \$ 50 ON YOUR INSURANCE BET**I HAVE BLACKJACK** -
MY HOLE CARD IS JACK OF CLUBS
YOU'RE EVEN

WAGER ?50

I SHOW 8 OF HEARTS
FIRST CARD IS KING OF HEARTS
NEXT CARD IS 9 OF DIAMONDS
HIT ?0
YOUR TOTAL IS 19
MY HOLE CARD IS 6 OF CLUBS
I DRAW 6 OF SPADES
MY TOTAL IS 20
YOU'RE BEHIND \$ 50

WAGER ?50

I SHOW 6 OF HEARTS
FIRST CARD IS 3 OF HEARTS
NEXT CARD IS 6 OF DIAMONDS
HIT ?1
NEXT CARD IS 3 OF CLUBS
HIT ?1
NEXT CARD IS 5 OF DIAMONDS
HIT ?0
YOUR TOTAL IS 17
MY HOLE CARD IS QUEEN OF CLUBS
I DRAW 7 OF HEARTS
I BUSTED***MY TOTAL IS 23
YOU'RE EVEN

WAGER ?50

I SHOW 5 OF CLUBS
FIRST CARD IS 10 OF HEARTS
NEXT CARD IS ACE OF HEARTS
BLACKJACK
MY HOLE CARD WAS 8 OF SPADES
YOU'RE AHEAD \$ 75

WAGER ?0

READY

*

This BASIC program simulates a lunar landing. The objective is to pilot the craft to a soft landing in a series of thrusts or burns, which decrease the rate of descent.

METHOD

The user is the pilot of the lunar module and is trying to land the craft on the surface of the moon. He inputs the amount of fuel to be burned each second of the descent. Printout is figures showing altitude, vertical velocity, and amount of fuel remaining after each burn.

INSTRUCTIONS

The program uses two files, one of which contains complete instructions for a lunar landing. A command within the program will produce this file listing. The program will also produce, upon command, a description of "How a Body Falls Onto the Moon."

If you have not played this game before, type INS as the first command, and read the detailed information provided.

SAMPLE PROBLEM

```
OLD OR NEW-OLD MOONER
READY
*
  RUN
```

FOR INSTRUCTIONS TYPE 'INS' AFTER 'COMMAND--?' APPEARS.

FOR STANDARD GAME TYPE 'STA' AFTER 'COMMAND--?' APPEARS.

COMMAND-- ?STX

STX

YOUR COMMAND IS ILLEGAL! LEGAL COMMANDS ARE:

NEW STA OLD INS EAR MOO TAB

TYPE O AT NEXT COMMAND--? FOR EXPLANATION OF COMMANDS

OR A LEGAL COMMAND TO CONTINUE

COMMAND-- ?O

O

COMMAND

EFFECT

INS - LISTS INSTRUCTIONS FOR OPERATING PROGRAM

STA - STANDARD INITIAL VALUES

NEW - NEW INITIAL VALUES

OLD - PREVIOUS INITIAL VALUES

EAR - EARTH LANDING

MOO - MOON LANDING

TAB - PRINTS TABLE OF HEIGHTS AND SPEEDS FOR FALLING BODY

COMMAND-- ?STA

STA

MOONER-2

STARTING HEIGHT: 500 FT
 STARTING SPEED: 50 FT/SEC
 FUEL SUPPLY: 120 UNITS
 MAXIMUM BURN: 30 UNITS/SEC
 BURN TO CANCEL GRAVITY OF MOON: 5 UNITS/SEC

CRASH TIME: 7.320508 SEC
 CRASH SPEED: 86.60254 FT/SEC

BURN TIME	HEIGHT	SPEED	FUEL
0	500	50	120
<u>78,8,8,8,8,8</u>			
1	451.5	47	112
8			
2	406	44	104
8			
3	363.5	41	96
8			
4	324	38	88
8			
5	287.5	35	80
8			
6	254	32	72
<u>77,7,7,7,7</u>			
7	223	30	65
7			
8	194	28	58
7			
9	167	26	51
7			
10	142	24	44
7			
11	119	22	37
<u>76,6,6,6</u>			
12	97.5	21	31
6			
13	77	20	25
6			
14	57.5	19	19
6			
15	39	18	13
<u>76,6,6,5</u>			
16	21.5	17	7
6			
17	5	16	1
6			
17.17	2.35	15.83	OUT OF FUEL. FREE FALL STARTS NOW.
0			
17.31	0	16.56	0

PLEASE DON'T LITTER.

WANT ANOTHER TRY WITH THIS DATA??-(YES OR NO)

?YES

BURN TIME	HEIGHT	SPEED	FUEL
0	500	50	120
<u>?8,8,8,8,8,8</u>			
1	451.5	47	112
8			
2	406	44	104
8			
3	363.5	41	96
8			
4	324	38	88
8			
5	287.5	35	80
8			
6	254	32	72
<u>?7,7,7,7,7</u>			
7	223	30	65
7			
8	194	28	58
7			
9	167	26	51
7			
10	142	24	44
7			
11	119	22	37
<u>?6,6,6,6,6,6</u>			
12	97.5	21	31
6			
13	77	20	25
6			
14	57.5	19	19
6			
15	39	18	13
6			
16	21.5	17	7
6			
17	5	16	1
<u>?1</u>			
17.3	0	17.2	.7

MAYBE YOU COULD GET A POOLED-RISK POLICY?

WANT ANOTHER TRY WITH THIS DATA??-(YES OR NO)

?NO

COMMAND-- ?END

This FORTRAN callable, GMAP subroutine allows a time-sharing program to use the time-sharing ACCESS system as a subroutine to perform functions relating to the file system.

INSTRUCTIONS

The calling sequence is:

```
CALL ACCESS (STRING,$SN)
or CALL ACCESS (STRING)
```

where STRING is an ASCII character string analogous to any short-form function response to the ACCESS system. The string can be a quoted literal and must in all cases conform to one of the following formats.

```
"# ... #"
" ... #"
" ... CR"
```

The system essentially deletes any leading #'s and replaces a trailing # with a carriage return.

\$SN represents an optional statement number to which control is transferred if the requested function cannot, for some reason, be performed.

METHOD

The basic technique employed by the ACCESS subroutine is to create a temporary file (*ACC), write the function description on this file, and invoke the ACCESS subsystem via a DRL CALLSS. Upon invocation, the ACCESS subsystem determines the presence of *ACC, obtains its input from it, and performs the indicated function. All keyboard input/output is eliminated when the ACCESS subsystem is used in this manner, except for the "LC" and "LS" functions. These functions produce the requested teletypewriter outputs.

Blanks appearing anywhere in the function description string are ignored and can be used freely to improve readability.

If an error is detected and an error return (\$SN) has not been specified, the subroutine issues the following message and terminate execution:

```
ACCESS SUBR. ERROR WITH NO ALTERNATE RETURN SPECIFIED
```

No provision is made to determine the type of error.

ACCESS-2

SAMPLE PROBLEM

```
CALL ACCESS ("AF,JDOE/CAT1$ABC/CAT2$AOK/FIL1,R#",$100)
CALL ACCESS ("CF,/FILEX,B/10,20/,R,W,MODE/RAND/#")
```

SAMPLE SOLUTION

The following example illustrates the use of this routine.

```
*NEW
READY
*10 CALL ACCESS("LS,/TTT#")
*20 STOP
*30 END
*RUN *=(ULIB)LIBRARY/APPLIB,R
```

```
FILE NAME-TTT
ORIGINATOR-EXOTIC
DATE CREATED-120172
DATE CHANGED-120472(15.80)
LAST DATE ACCESSED-120472
MAX FILE SIZE- 20 BLOCKS
CURRENT FILE SIZE- 1 BLOCKS
FILE TYPE-LINKED (ASCII)
DEVICE-DS4
GENERAL PERMISSIONS-NONE
SPECIFIC PERMISSION-
NONE
```


This FORTRAN-callable GMAP subroutine enables a user program to determine if execution is in time sharing or batch, and if the internal character code is ASCII or BCD.

INSTRUCTIONS

The calling sequence for this routine is:

```
CALL APARAM (I, J)
```

where

- On input, if I=4, J is set to 0 if in batch mode. If in time sharing mode, J is set to non-0.
- If I=5, J is set to 0 if BCD. If ASCII, J is set to non-0.

The source version of this routine cannot be used directly, however, the object version is contained in the random library APPLIB. A program can use this routine by referencing APPLIB as a user library when compiling and loading.

SAMPLE RUN

The following example illustrates the use of this routine.

```
*NEW
READY
*10 CALL APARAM(4,J)
*20 CALL APARAM(5,K)
*30 PRINT, J, K
*40 STOP
*50 END
*RUN *=(ULIB)LIBRARY/APPLIB,R
          1          -1

*RUN *=(ULIB,BCD)LIBRARY/APPLIB,R
          1          0

*
```


The file APPLIB is a user's random library containing a number of FORTRAN-callable subroutines. The specific user instructions for each of these routines is documented under the name of the source file for that routine. The file APPLIB-R is the R* version of APPLIB. This file is required if APPLIB is to be modified, augmented, or recreated.

INSTRUCTIONS

If a FORTRAN program references a routine on APPLIB, then, when the programs are bound to form the run file, the file APPLIB must be referenced as a user's random library. In time sharing this is done by typing the RUN command using a format similar to that below:

```
RUN source = runfile (ULIB) LIBRARY/APPLIB, R
```

In batch processing it is done by including cards similar to the following:

```
$ LIBRARY AP
$ PRMFL AP, READ, RANDOM, LIBRARY/APPLIB
```

NOTE: Some of the routines contained in APPLIB use system modules that may not yet be available if your system is not on the latest software release. However, versions of the required system modules are also included in APPLIB. The APPLIB versions of these routines will be used if a special CALL statement is executed before executing any of the routines requiring these system modules. These special CALL statements indicate the software release version implemented on your system. They take one of the following forms:

```
CALL SRB4("ABC")
CALL SRC5("ABC")
CALL SRD6
CALL SRE7
CALL SRF8
CALL SRAPn (n=0, 1, ...)
```

In SRB4 and SRC5, the parameter "ABC" is used to set the internal ASCII/BCD flag. The routines SRAPn are used for internal linking within APPLIB and to load some special purpose functions.

ROUTINES IN APPLIB

Currently, the following routines are contained in APPLIB. The specific documentation for each of these routines contains user instructions and examples. If the routine requires system modules not available under earlier software releases, the minimum system release level required to execute the routine without including a special CALL SRxx is indicated in parenthesis:

APPLIB-2

ACCESS		
APARAM	(SR-D/6)	DEFIL
ASCB CD		KIN
BCDASC		UATOLA
CALLSS		

MODIFYING APPLIB

The system maintenance group at a user site can modify the file APPLIB-R by submitting a FILEDIT job similar to the following:

```
$ IDENT
$ USERID      LIBRARY $ password
$ FILEDIT     SOURCE,OBJECT,UPDATE, NONE
$ PRMFL       *R, READ, SEQ, LIBRARY/APPLIB-R
$ FILE        R*, X1S, 2L
$ DATA       M*, , COPY
$ ENDEDIT
$ ENDCOPY
$ DATA       *C,, COPY
filedit directives
$ ENDCOPY
$ UTILITY
$ FILE        IN, X1R
$ PRMFL       R*, WRITE, SEQ, LIBRARY/APPLIB-R
$ FUTIL       IN, R*, RWD/IN, R*/ , COPY/1F/
$ ENDJOB
```

The file APPLIB can be recreated from the file APPLIB-R by submitting a job similar to the following:

```
$ IDENT
$ USERID      LIBRARY $ password
$ PROGRAM     RANLIB
$ PRMFL       R*, READ, SEQ, LIBRARY/APPLIB-R
$ PRMFL       A4, READ/WRITE, RANDOM, LIBRARY/APPLIB
$ ENDJOB
```

This FORTRAN-callable GMAP subroutine converts a character string from 9-bit ASCII code to 6-bit BCD code. This subroutine may be used in batch or time-sharing mode.

INSTRUCTIONS

The calling sequence for this routine is:

```
CALL ASCBCD (IN,OT,ICOUNT)
```

where

- IN is the character to be converted.
- ICOUNT is the number of characters to be converted.
- OT is the character array where the converted characters are to be stored.

IN is assumed to contain four characters per machine word. OT will be packed six characters per machine word.

The source version of this subroutine cannot be used directly; however, the object version is contained in the random library APPLIB. A program can use this subroutine by referencing APPLIB as a user library when compiling and loading.

The following chart displays the BCD-to-ASCII character translation.

ASCB CD-2

BCD-to-ASCII CHARACTER TRANSLATION

ASCII	BCD	ASCII	BCD	ASCII	BCD	ASCII	BCD
000	32 &	040 Ø	20 Ø	100 @	14 @	140 \	57 \
001	32 &	041 !	77 !	101 A	21 A	141 a	21 A
002	32 &	042 "	76 "	102 B	22 B	142 b	22 B
003	32 &	043 #	13 #	103 C	23 C	143 c	23 C
004	32 &	044 \$	53 \$	104 L	24 D	144 d	24 D
005	32 &	045 %	74 %	105 E	25 E	145 e	25 E
006	32 &	046 &	32 &	106 F	26 F	146 f	26 F
007	32 &	047 /	57 /	107 G	27 G	147 g	27 G
010	32 &	050 (	35 (	110 H	30 H	150 h	30 H
011	32 &	051)	55)	111 I	31 I	151 i	31 I
012	32 &	052 *	54 *	112 J	41 J	152 j	41 J
013	32 &	053 +	60 +	113 K	42 K	153 k	42 K
014	32 &	054 ,	73 ,	114 L	43 L	154 l	43 L
015	20 Ø	055 -	52 -	115 M	44 M	155 m	44 M
016	32 &	056 .	33 .	116 N	45 N	156 n	45 N
017	32 &	057 /	61 /	117 O	46 O	157 o	46 O
020	32 &	060 0	00 0	120 P	47 P	160 p	47 P
021	32 &	061 1	01 1	121 Q	50 Q	161 q	50 Q
022	32 &	062 2	02 2	122 R	51 R	162 r	51 R
023	32 &	063 3	03 3	123 S	62 S	163 s	62 S
024	32 &	064 4	04 4	124 T	63 T	164 t	63 T
025	32 &	065 5	05 5	125 U	64 U	165 u	64 U
026	32 &	066 6	06 6	126 V	65 V	166 v	65 V
027	32 &	067 7	07 7	127 W	66 W	167 w	66 W
030	32 &	070 8	10 8	130 X	67 X	170 x	67 X
031	32 &	071 9	11 9	131 Y	70 Y	171 y	70 Y
032	32 &	072 :	15 :	132 Z	71 Z	172 z	71 Z
033	32 &	073 ;	56 ;	133 [	12 [	173 (	32 &
034	32 &	074 <	36 <	134 \	37 \	174 !	32 &
035	32 &	075 =	75 =	135]	34 J	175)	32 &
036	32 &	076 >	16 >	136 ^	40 ^	176 ~	32 &
037	32 &	077 ?	17 ?	137 _	20 ←	177 DEL	32 &

SAMPLE RUN

The following run illustrates the use of the ASCBCD subroutine.

*LIST

```

10 INTEGER K(3)
20 INTEGER L(2)
30 10 PRINT,"ENTER DECIMAL VALUE OF AN ASCII CHARACTER"
40 READ,K(3)
50 IF(K(3).LT.0) STOP
60 CALL ASCBCD(K,L,12)
70 PRINT 20,L(2)
80 20 FORMAT(" THE OCTAL VALUE OF THE BCD EQUIVALENT IS ",02)
90 GO TO 10
100 END

```

READY

*RUN *(ULIB)LIBRARY/APPLIB,R

ENTER DECIMAL VALUE OF AN ASCII CHARACTER

=48

THE OCTAL VALUE OF THE BCD EQUIVALENT IS 00

ENTER DECIMAL VALUE OF AN ASCII CHARACTER

=55

THE OCTAL VALUE OF THE BCD EQUIVALENT IS 07

ENTER DECIMAL VALUE OF AN ASCII CHARACTER

=07

THE OCTAL VALUE OF THE BCD EQUIVALENT IS 32

ENTER DECIMAL VALUE OF AN ASCII CHARACTER

=-1

*

ASCII		BCD	ASCII	BCD	ASCII	BCD	ASCII	BCD
000	NULL	17 ?	040 ¢	20 ¢	100 @	14 @	140 `	17 ?
001	SOH	17 ?	041 !	77 !	101 A	21 a	141 a	21 a
002	STX	17 ?	042 "	76 "	102 B	22 b	142 b	22 b
003	ETX	17 ?	043 #	13 #	103 C	23 c	143 c	23 c
004	EOT	17 ?	044 \$	53 \$	104 D	24 d	144 d	24 d
005	ENG	17 ?	045 %	74 %	105 E	25 e	145 e	25 e
006	ACK	17 ?	046 &	32 &	106 F	26 f	146 f	26 f
007	BELL	17 ?	047 '´	57 '´	107 G	27 g	147 g	27 g
010	BSP	17 ?	050 (	35 (	110 H	30 h	150 h	30 h
011	HT	17 ?	051)	55)	111 I	31 i	151 i	31 i
012	LF	17 ?	052 *	54 *	112 J	41 j	152 j	41 j
013	VT	17 ?	053 +	60 +	113 K	42 k	153 k	42 k
014	FFD	17 ?	054 ,	73 ,	114 L	43 l	154 l	43 l
015	CR	17 ?	055 -	52 -	115 M	44 m	155 m	44 m
016	SO	17 ?	056 .	33 .	116 N	45 n	156 n	45 n
017	SI	17 ?	057 /	61 /	117 O	46 o	157 o	46 o
020	DLE	17 ?	060 0	00 0	120 P	47 p	160 p	47 p
021	DC1	17 ?	061 1	01 1	121 Q	50 q	161 q	50 q
022	DC2	17 ?	062 2	02 2	122 R	51 r	162 r	51 r
023	DC3	17 ?	063 3	03 3	123 S	62 s	163 s	62 s
024	DC4	17 ?	064 4	04 4	124 T	63 t	164 t	63 t
025	NAK	17 ?	065 5	05 5	125 U	64 u	165 u	64 u
026	SYN	17 ?	066 6	06 6	126 V	65 v	166 v	65 v
027	ETB	17 ?	067 7	07 7	127 W	66 w	167 w	66 w
030	CAN	17 ?	070 8	10 8	130 X	67 x	170 x	67 x
031	EM	17 ?	071 9	11 9	131 Y	70 y	171 y	70 y
032	SUB	17 ?	072 :	15 :	132 Z	71 z	172 z	71 z
033	ESC	17 ?	073 ;	56 ;	133 [	12 [	173 }	12 [
034	FS	17 ?	074 <	36 <	134 \	37 \	174 ;	17 ?
035	GS	17 ?	075 =	75 =	135]	34]	175 }	34]
036	RS	17 ?	076 >	16 >	136 ^	40 ^	176 ~	17 ?
037	US	17 ?	077 ?	17 ?	137 _	72 _	177 DEL	17 ?

This FORTRAN-callable GMAP subroutine converts a character string from six-bit BCD code to nine-bit upper case ASCII code. This subroutine may be used in either time-sharing or batch mode.

INSTRUCTIONS

The calling sequence for this routine is:

```
CALL BCDASC (IN,OT,ICOUNT)
```

where

- IN is the character array to be converted.
- ICOUNT is the number of characters to be converted.
- OT is the character array where the converted characters are to be stored.

IN assumed to contain six characters per machine word. OT will be packed four characters per machine word. IN and OT can be in the same array.

The source version of this subroutine cannot be used directly; however, the object version is contained in the random library APPLIB. A program can use this subroutine by referencing APPLIB as a user library when compiling and loading.

The following chart displays the BCD-to-ASCII character translation.

6-bit	CHAR	9-bit	6-bit	CHAR	9-bit	6-bit	CHAR	9-bit
00	0	060	25	E	105	52	-	055
01	1	061	26	F	106	53	\$	044
02	2	062	27	G	107	54	*	052
03	3	063	30	H	110	55	)	051
04	4	064	31	I	111	56	;	073
05	5	065	32	&	046	57	'	047
06	6	066	33	.	056	60	+	053
07	7	067	34	]	135	61	/	057
10	8	070	35	(	050	62	S	123
11	9	071	36	<	074	63	T	124
12		133	37	\	134	64	U	125
13	#	043	40	↑	136	65	V	126
14	@	100	41	J	112	66	W	127
15	:	072	42	K	113	67	X	130
16	>	076	43	L	114	70	Y	131
17	?	077	44	M	115	71	Z	132
20	␣	040	45	N	116	72	-	137
21	A	101	46	O	117	73	,	054
22	B	102	47	P	120	74	%	045
23	C	103	50	Q	121	75	=	075
24	D	104	51	R	122	76	"	042
						77	!	041

BCDASC-2

SAMPLE RUN

The following run illustrates the use of this subroutine.

*LIST

```
10 CHARACTER L*4(3)
20 INTEGER K(2)
30 10 PRINT,"ENTER DECIMAL VALUE OF A BCD CHARACTER"
40 READ,K(2)
50 IF(K(2).LT.0) STOP
60 CALL BCDASC(K,L,12)
70 PRINT 20,L(3),L(3)
80 20 FORMAT(" THE ASCII EQUIVALENT IS ",R1," (OCTAL="",33,"")")
90 GO TO 10
100 END
```

READY

```
*RUN *=(ULIB)LIBRARY/APPLIB,R
ENTER DECIMAL VALUE OF A BCD CHARACTER
=12
THE ASCII EQUIVALENT IS 0 (OCTAL=100)
ENTER DECIMAL VALUE OF A BCD CHARACTER
=49
THE ASCII EQUIVALENT IS / (OCTAL=057)
ENTER DECIMAL VALUE OF A BCD CHARACTER
=-1
```

This FORTRAN-callable GMAP subroutine calls a time sharing subsystem and will return to the calling program. The called subsystem can be any subsystem known to the TSS executive.

INSTRUCTIONS

The calling sequence for this routine is:

```
CALL CALLSS (STRING)  or
CALL CALLSS (STRING, NAME)
```

where

- NAME is the four-character ASCII name of the subsystem to be called.
- STRING is the command sent to invoke the system in normal mode. STRING is an ASCII character (constant or variable) that must contain a carriage return or # as a terminating character. If the terminating character is # it will be replaced by a carriage return when CALLSS is called. If NAME is not supplied, the first four characters in STRING are used for NAME.

For example, in the normal time-sharing mode, the following command line causes the specific attributes of a file to be printed:

```
CATALOG  FILENAME
```

A FORTRAN program can cause the same printout to occur by using the following call:

```
CALL CALLSS ("CATALOG  FILENAME#")
```

The special string termination character, #, can be changed to any other character by calling FIXCR:

```
CALL FIXCR (CHAR)
```

where CHAR contains the octal equivalent of the new termination character, right justified.

The source version of this routine cannot be used directly, however, the object version is contained in the random library APPLIB. A program can use this routine by referencing APPLIB as a user library when compiling and loading.

RESTRICTIONS

Nesting to more than two levels using CALLSS is not permitted. If the called time sharing system is SYSTEM, then control will not be returned to the calling program.

CALLSS-2

SAMPLE RUN

The following example illustrates the use of this subroutine.

```
*NEW  
READY  
*10 PRINT,"START"  
*20 CALL CALLSS("STATF#")  
*30 CALL CALLSS("REM0 APPLIB#")  
*40 CALL CALLSS("STATF#")  
*50 PRINT,"STOP"  
*60 STOP  
*70 END  
*RUN *=(ULIB)LIBRARY/APPLIB,R  
START
```

LIST OF OPEN FILES: APPLIB SYSLIB

LIST OF OPEN FILES: SYSLIB

STOP

This FORTRAN subroutine transfers characters from one character string variable, starting at a specified position, to a second character string variable, until a delimiter character is found.

INSTRUCTIONS

The calling sequence is:

```
CALL DCS(DEL,ND,MD,NPOINT,INSTR,MAXIN,OTSTR,MAXOT)
```

where the character variables DEL, INSTR and OTSTR are defined as:

```
CHARACTER DEL*1(ND), INSTR*MAXIN, OTSTR*MAXOT
```

Starting at the NPOINT character of INSTR, the routine transfers characters into OTSTR until MAXOT characters have been transferred or one of the delimiting characters DEL is found. In either case, NPOINT will be advanced to the character following the next delimiter. MD is set so that the delimiter causing termination of the character transfer was DEL(MD); if MD = 0, then the end of INSTR was found before a delimiter.

SAMPLE RUN

The following example illustrates the use of this subroutine.

*LIST

```
010 CHARACTER A*30, B*30
020 CHARACTER DEL*1(2)/",", ";"/
030 PRINT, "ENTER A STRING"
040 READ, A
050 NPOINT = 1
060 10 B = " "
070 CALL DCS(DEL,2,MD,NPOINT,A,30,B,30)
080 PRINT, "NPOINT =", NPOINT
090 IF(MD .EQ. 0) GO TO 20
100 PRINT, "DELIMITER FOUND WAS ", DEL(MD)
110 PRINT, B
120 GO TO 10
130 20 PRINT, "END OF INPUT LINE"
140 PRINT, B
150 STOP
160 END
```

DCS-2

READY

*RUN *(ULIB)LIBRARY/APPLIB,R

ENTER A STRING

= "ABCD,EFG,H,IJKLMN"

NPØINT = 6

DELIMITER FOUND WAS ,

ABCD

NPØINT = 10

DELIMITER FOUND WAS ,

EFG

NPØINT = 12

DELIMITER FOUND WAS ,

H

NPØINT = 31

END OF INPUT LINE

IJKLMN

*

This FORTRAN-callable GMAP subroutine creates a named temporary file and accesses it in the user's available file table.

INSTRUCTIONS

The calling sequence is:

```
CALL DEFIL (NAME, LINKS, MODE, ISTAT)
```

where

- NAME is a character *8 variable containing the ASCII name of the temporary file to be created.
- LINKS is the size in links at which the file is to be created. If MODE=0, a sequential file is created. If MODE#0, a random file is created, ISTAT contains the status indication shown below:

ISTAT =	0	Successful
	3	No room in AFT
	4	Temporary file not available
	5	Duplicate file name
	6	No room in PAT

NOTE: This routine does not make the proper linkages to allow FORTRAN IO to read from or write to the file. The source version of this routine cannot be used directly; however, the object version is contained in the random library APPLIB. A program can use this routine by referencing APPLIB as a user library when compiling and loading.

SAMPLE RUN

The following example illustrates the use of this routine.

```
*NEW  
READY
```

```
*10 CALL DEFIL("TEST ",1,0,ISTAT)  
*20 PRINT,"ISTAT=",ISTAT  
*30 STØP  
*40 END  
*STATF
```

```
LIST OF OPEN FILES: NONE
```

```
*RUN ** (ULIB) LIBRARY/APPLIB, R  
ISTAT= 0
```

```
*STATF
```

```
LIST OF OPEN FILES: APPLIB TEST
```

```
*
```


This FORTRAN program provides a time sharing interface to the GMAP assembler. The file GMAP-SOR is the source version of the program. The file GMAP is the H* version, the execution of which is initiated using the command loader.

METHOD

The program passes a time sharing file to the batch GMAP assembler. When the batch portion of the program has terminated, the P* and C* files written by the assembler are available to the time sharing user.

INSTRUCTIONS

The execution of this program should be invoked using the command loader by typing a command line similar to:

LIBRARY/GMAP or

LIBRARY/GMAP: option = value, option = value, etc.

The legal options and their possible values follow. If any option is not specified, the default value is used.

SOURCE = filename	Allows the user to specify the GMAP source file to be assembled. This file should be in standard CARDIN format. If first line formatting information is not contained in the file, the user will be asked for label disposition and tab characters and settings. If a SOURCE file is not specified, the current file will be assumed.
P*=filename	Allows the user to specify the P* file to be used for the assembly listing. If the file is not already in the AFT, a temporary file by the specified name will be created and used. If a P* filename is not specified, the filename PSTAR will be assumed.
C*=filename	Allows the user to specify the C* file to be used for the object deck. If the file is not already in the AFT, a temporary file by the specified name will be created and used. If a C* filename is not specified, the filename CSTAR will be assumed.
REMOVE= $\left\{ \begin{array}{l} \text{TRUE} \\ \text{FALSE} \end{array} \right\}$	Sets indicators to remove temporary working files when no longer needed. The default value is TRUE.
URGC = nn	Sets the batch urgency to the value specified (1-40). 40 is the default value.
CORE = nn	Sets the batch core limits to the value indicated (in K). The default value is 24.

The following sample commands illustrate the correct use of the option fields:

```
LIBRARY/GMAP:
LIBRARY/GMAP:SOURCE=PROG1
LIBRARY/GMAP:SOURCE=PROG2,C*=OBJ
LIBRARY/GMAP:C*=OBJ,REMOVE=FALSE
LIBRARY/GMAP:URGC=5,CORE=18
```

SAMPLE RUN

The following printout illustrates the use of this program to perform a GMAP assembly as though the GMAP assembler were a time-sharing routine.

```
*NEW
READY
*10##NORM
*20:ITL:TEST GMAP PROGRAM
*30:LBL:TEST
*40:SYMDEF:ENTER
*50ENTER:NULL
*60:EAX0:=4,DU
*70:LDA:-5,0
*80:TRA:5,AL
*90:END
*LIBRARY/GMAP:
```

*STATF

LIST OF OPEN FILES: PSTAR CSTAR

```
*SCAN PSTAR
FORM?GMAP
001 ERRORS
```

```
EDIT?YES
?ERRORS
X 000000 000004 6200 03 000 5 EAX0 =4,DU 60 #0021
?P ALL
#0001
2122T 01 02-21-73 10.602 TEST GMAP PROGRAM PAGE 1 #0002
PREFACE #0003
PROGRAM BREAK 6 #0004
COMMON LENGTH 0 #0005
V COUNT BITS 5 #0006
PRIMARY SYMDEF ENTRY #0007
ENTER 0 #0008
#0009
SECONDARY SYMDEF ENTRY #0010
#0011
BLOCK LENGTH #0012
SYMREF #0013
END OF BINARY CARD TEST0001 #0014
```

```

#0015
2122T 01 02-21-73 10.602 TEST GMAP PROGRAM PAGE 2 #0016
1 TTL TEST GMAP PROGRAM 20 #0017
2 LBL TEST 30 #0018
3 SYMDEF ENTER 40 #0019
000000 4 ENTER NULL 50 #0020
X 000000 000004 6200 03 000 5 EAX0 =4,DU 60 #0021
000001 777773 2350 10 000 6 LDA -5,0 70 #0022
000002 000005 7100 05 000 7 TRA 5,AL 80 #0023
#0024
ERROR LINKAGE #0025
000003 000000000000 000 #0026
000004 254563255120 000 #0027
END OF BINARY CARD TEST0002 #0028
8 END 90 #0029
6 IS THE NEXT AVAILABLE LOCATION. #0030
GMAP VERSION/ASSEMBLY DATES JMPA 091572/091472 JMPB 053172/070872
JMPC 072772/072772 #0031
THERE WERE 1 WARNING FLAGS IN THE ABOVE ASSEMBLY #0032
ON PAGE NO. #0033
2 #0034
#0035
2122T 01 02-21-73 10.602 TEST GMAP PROGRAM PAGE 3 #0036
OCTAL SYMBOL REFERENCES BY ALTER NO. #0037
0 ENTER 4 3 4 #0038
** 19K LIMITS NEEDED FOR THIS ASSEMBLY. #0039
EOF

?DONE
*SCAN CSTAR
FORM?USER
CODE?
EDIT?YES
?P ALL
$ OBJECT G10.602022173TEST00000000 #0001
$ DKEND TEST00030000 #0002
EOF

```

?DONE

*

This FORTRAN-callable, GMAP subroutine reads the last line again from the terminal input buffer.

INSTRUCTIONS

The calling sequence for this routine is:

```
CALL KIN (STRING, ICOUNT)
```

where

- STRING contains the last line in the terminal input buffer.
- ICOUNT is the number of characters stored in STRING. If ICOUNT=0, the buffer was empty. STRING should be defined as an ASCII character variable large enough to hold the last input line.

The source version of this routine cannot be used directly; however, the object version is contained in the random library APPLIB. A program can use this routine by referencing APPLIB as a user library when compiling and loading.

SAMPLE PROBLEM

The command loader leaves data following a colon (:) on the command line which invoked it in the input buffer. Write a sample program which, when invoked by the command loader, will retrieve any information following the colon:

SAMPLE RUN

```
*NEW
*10 CHARACTER STRING*72
*20 CALL KIN(STRING,ICOUNT)
*30 IF(ICOUNT),20,
*40 PRINT,"DATA RETRIEVED IS"
*50 PRINT,STRING
*60 STOP
*70 20 PRINT,"BUFFER WAS EMPTY"
*80 STOP
*90 END
*RUN *=TEST(NOG0,ULIB)LIBRARY/APPLIB.R
*/TEST:SAMPLE DATA
DATA RETRIEVED IS
SAMPLE DATA
```


These FORTRAN-callable, GMAP subroutines convert an ASCII character string from upper to lower case or from lower to upper case. These subroutines may be used in either time sharing or batch modes.

INSTRUCTIONS

The calling sequence for converting to lower case is:

```
CALL  UATOLA (STRING, ICOUNT)
```

The calling sequence for converting to upper case is:

```
CALL  LATOUA (STRING, ICOUTN)
```

where

- ICOUNT is the number of characters to be converted. The subroutine assumes the characters are packed four per machine word.
- STRING, on input, is the character string to be converted, and on output, is the converted character string.

The source version of these subroutines cannot be used directly, however the object version is contained in the random library APPLIB. A program can use any of these subroutines by referencing APPLIB as a user library when compiling and loading.

SAMPLE RUN

The following example illustrates the use of the UATOLA subroutines.

```
*NEW  
READY  
*10 CHARACTER A*12/"AABBCCDD12()"  
*20 PRINT,A  
*30 CALL UATOLA(A,12)  
*40 PRINT,A  
*50 CALL LATOUA(A,12)  
*60 PRINT,A  
*70 STOP  
*80 END  
  
#RUN *=(ULIB)LIBRARY/APPLIB,R  
AABBCCDD12()  
AABBCCDD12()  
AABBCCDD12()
```